1-1-2003

Enhancements of CINET fuzzy classifier

Zhen Yu
Iowa State University

Enhancements of CINET fuzzy classifier

by

Zhen Yu

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Electrical Engineering

Program of Study Committee:
Ratnesh Kumar (Major Professor)
Julie Dickerson
Partha Sarkar

Iowa State University

Ames, Iowa

2003

Graduate College
Iowa State University

This is to certify that the Master's thesis of

Zhen Yu

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my major professor, Dr. Ratnesh Kumar, for his kind guidance, support, and encouragement during my graduate study, and taking time to revise my thesis. I also thank Dr. James A. Stover of Applied Research Lab, Pennsylvania State University, for him to provide many stimulating ideas for the research undertaken in this thesis. I would sincerely like to convey my appreciation to the committee members, Dr. Julie Dickerson and Dr. Partha Sarkar for their time and input.

My deep gratitude goes to my wife, Min Xu, for her warm love, support, and encouragement during my life. I am also thankful to my parents in China and all my friends in ISU for their suggestions and kind help during my study.

The research was supported in part by the National Science Foundation under the grants NSF-ECS-9709796, NSF-ECS-0099851, NSF-ECS-0218207, NSF-ECS-0244732, a DoD-EPSCoR grant through the Office of Naval Research under the grant N000140110621, a KYDEPSCoR grant, and a grant from Iowa State University.

ABSTRACT

Neuro-fuzzy systems combine the theory of two popular computational intelligence techniques: neural networks and fuzzy logic systems. In this thesis, we study Continuous Inference Network (CINET), a neuro-fuzzy classifier, developed at Applied Research Lab, Pennsylvania State University.

Our work is to make some enhancements of CINET classifier. We prove that the problem of learning the range of input membership function of CINET classifier is a Linear Mixed Integer Programming (LMIP) problem. Moreover, the necessity and sufficiency functions for fuzzy inference are simplified to satisfy some algebraic properties and facilitate using back-propagation algorithm to adjust system parameters. To deal with the randomness in input measurements we also define the ambiguity degree and give out its calculation method.

CHAPTER 1. INTRODUCTION

Designing a classification system is a kind of supervised learning problem, which is used to estimate an unknown input-output mapping from the known input-output samples. Example input-output samples are necessary for training in supervised learning. In contrast, clustering belongs to the other class of learning problem, namely unsupervised learning, in which only input samples are given and there is no notion of the output.

Fuzzy systems and neural networks are two popular methods to approach supervised learning problems. They both can be used to design a classifier. Although functionally similar, they differ in the way that they estimate input-output functions, and represent and encode knowledge about input-output functions.

Generally speaking, fuzzy classification system consists of a set of fuzzy if-then rules and generates output with respect to the given inputs through fuzzy inference operations. Fuzzy if-then rules can encode the knowledge from human expert and they are interpretable and easy to understand by human. Wang [33] proved that fuzzy systems are capable of approximating any nonlinear continuous function on a compact set to arbitrary accuracy. Neural network is made up of a large number of highly interconnected simple computing node elements. It is well known that neural network also has the ability to approximate any nonlinear input-output function [5].

Fuzzy system theory lacks a precise guideline for the choice of membership functions, fuzzy operators, and reasoning schemes. Neural network theory suffers from unstructured knowledge representation and its inability to evaluate the amount of learning efficiency. Therefore, it is natural to integrate the two together to utilize the advantages of both.

So far various applications have been reported for the integration of fuzzy systems and neural networks, often called neuro-fuzzy system or fuzzy neural network, first proposed by Lee and Lee [19] in 1975. Horikawa etc. [6] presented a fuzzy modeling network (FMN) method by using fuzzy networks with the back-propagation algorithm. Wang [32] proposed a fuzzy basis function network (FBFN) and used the orthogonal least-squares algorithm to tune consequent parameters. Jang [8] also proposed an adaptive network-based fuzzy inference system (ANFIS). All these applications employ the structure of neural network to represent fuzzy if-then rules and use some learning algorithms for parameter tuning.

CINET (Continuous Inference Network), proposed by Stover and Gibson [29] of Applied Research Laboratory at Pennsylvania State University, is a multi-node fuzzy classification system. Each node in CINET is a aggregator that generates a confidence factor for the existence of some property from input data. Node inputs are a set of sub-properties with varying confidences of existence and varying degrees of significance to the output property. Typical node aggregation functions are mathematical models of "AND", "OR" and "NOT" functions. CINET and its fuzzy calculus have been successfully used in various fields, such as process control and medical diagnosis.

The problems of fuzzy classification can be divided into two parts: structure identification and parameter identification. The former is related to finding a suitable rule providing a proper partition of the input space. On the other hand, the latter deals with the adjustment of classification system parameters. In this work, we propose some enhancements of CINET fuzzy classifier according to these two aspects.

As to structure identification, we show that the input space partition problem in CINET could be formulated as a Linear Mixed Integer Programming (LMIP) problem, which

can be solved by optimization algorithm or optimization software such as ILOG CPLEX [7]. Moreover, we suggest some modified functions for fuzzy-aggregation operations, which are simpler and better algebraically behaved. These simplified functions are continuous, so they also facilitate the use of back-propagation algorithm for parameter adjustment. The weight of each connection in CINET is significance degree of each input property or internal combination of input properties. We show that the weight can be modified by variance of sensor data to deal with measurement randomness of sensor signal.

The rest of thesis is organized as follows. Chapter 2 introduces the principles of fuzzy logic theory. Chapter 3 discusses neuro-fuzzy system and introduces CINET classifier. Chapter 4 describes the enhancements of CINET, including determining the range of membership function of input variables, simplifying fuzzy connective functions, and computing ambiguity degree to deal with measurement randomness. Chapter 5 presents the conclusions, and discusses the direction of future work.

CHAPTER 2. PRINCIPLE OF FUZZY LOGIC

Fuzzy logic is a logical system providing a mathematical framework to capture the uncertainties associated with human cognitive systems such as thinking and reasoning. Simply speaking, it simulates human thinking, which operates more likely on symbols than exact values. In fact, our daily thoughts and communication are full of these symbols or fuzzy expressions.

Fuzzy sets and membership functions

Fuzzy sets

Fuzzy set was proposed by Lotfi Zadeh in 1965 as a numerical means to handle the uncertainty and vagueness inherent to human perception, speech, thinking and so on. The most straightforward example is the linguistic uncertainty of human language. Let us consider a linguistic variable "warm" with respect to the temperature of weather. Most people may agree that 65° F is warm, but how about 75° F? Someone may agree it is warm, but others may think it is hot. Different people have different answers. Therefore, we find it is very hard to decide a boundary value that can absolutely differentiate between "warm" and "hot". This is due to the linguistic ambiguity of our language, which motivates the proposing of fuzzy set to deal with this ambiguity.

A classical set is a set with a crisp boundary. For example, a classical set A can be expressed as:

$$A = \{x \mid 70 \geq x \geq 60,\ x \in R\}, \tag{2.1.1}$$

where there are two unambiguous boundary values 60 and 70, such that any x value between them belongs to the set A, otherwise not.

Fuzzy set and membership function: If X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\}, \tag{2.1.2}$$

$$\mu_A : X \rightarrow [0, 1], \tag{2.1.3}$$

where μ_A, called membership function, is a mapping from X to a real value between 0 and 1. $\mu_A(x)$ expresses the degree to which x belongs to the set A.

A classical set, which is often called crisp set in the fuzzy literature crisp set, can also be associated with a membership function:

$$\mu_A : X \rightarrow \{0, 1\}, \tag{2.1.4}$$

where $\mu_A(x)$ is one if x belongs to the set A, and zero, otherwise.

An alternative way of denoting a fuzzy set A is:

$$A = \begin{cases} \sum_{x_i \in X} \mu_A(x_i) / x_i, & \text{if } X \text{ is discrete.} \\ \int_X \mu_A(x) / x, & \text{if } X \text{ is continuous.} \end{cases} \tag{2.1.5}$$

The summation and integration signs in (2.1.5) stand for the union of $(x, \mu_A(x))$ pairs. Similarly, "/" is only a marker and does not imply division.

Various membership functions

In case of discrete X, we can discretely assign membership degree for each element of the set A. However, for fuzzy set with continuous X, we should define a continuous membership function. Various functions can be chosen as membership function. Some

widely used are triangular, trapezoidal and Gaussian functions, which are depicted in Figure 2.1, and their mathematical forms appear in Table 2.1.

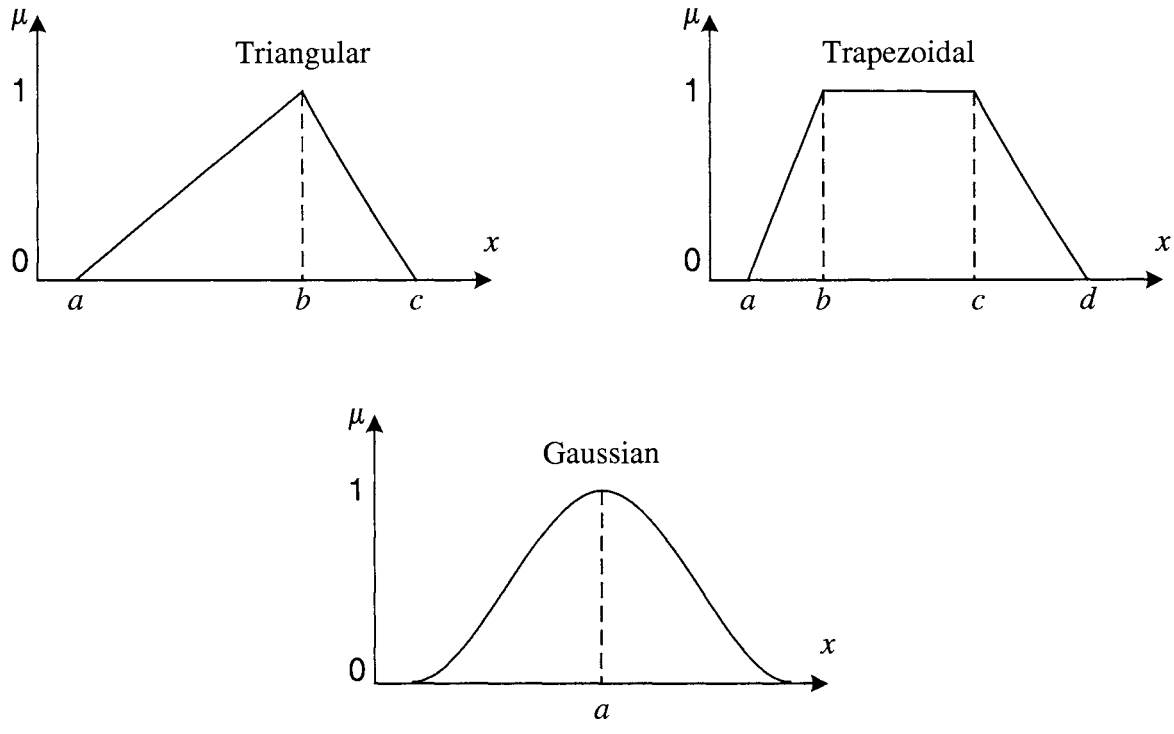


Figure 2.1 Triangular, trapezoidal and Gaussian membership functions

Table 2.1 Membership function formulas

Triangular:	$\mu(x) = \begin{cases} \frac{x-a}{b-a}, & \text{if } x \in (a, b], \\ \frac{c-x}{c-b}, & \text{if } x \in (b, c], \\ 0, & \text{otherwise.} \end{cases}$
Trapezoidal:	$\mu(x) = \begin{cases} \frac{x-a}{b-a}, & \text{if } x \in (a, b], \\ 1, & \text{if } x \in (b, c], \\ \frac{d-x}{d-c}, & \text{if } x \in (c, d], \\ 0, & \text{otherwise.} \end{cases}$
Gaussian:	$\mu(x) = exp\{-\frac{(x-a)^2}{2\sigma^2}\}.$

As to the set "warm" mentioned above, we can assign membership degree as in Figure 2.2 for crisp set and fuzzy set. For crisp set all temperature values from 60° F to 70° F belong to set "warm" with membership degree one, however, fuzzy set gives out a smoother membership degree curve and assigns different degree to different temperature value.

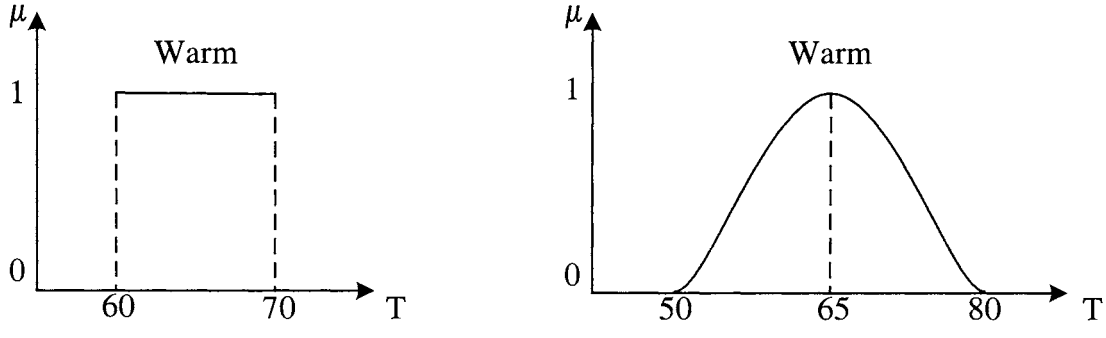


Figure 2.2 Example of crisp set and fuzzy set

Operations on fuzzy sets

Intersection, union and complement

The logical operations, such as intersection, union and complement, on fuzzy sets are generalizations of those on crisp sets. Typically, minimum and maximum are used for intersection and union operations. Let A and B be two fuzzy sets on the same universe of discourse X. Then, their intersection, union and complement are as defined below, and depicted in Figure 2.2.

Intersection: $$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, \quad \forall x \in X. \tag{2.2.1}$$

Union: $$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}, \quad \forall x \in X. \tag{2.2.2}$$

Complement: $$\mu_{\bar{A}}(x) = 1 - \mu_A(x), \quad \forall x \in X. \tag{2.2.3}$$

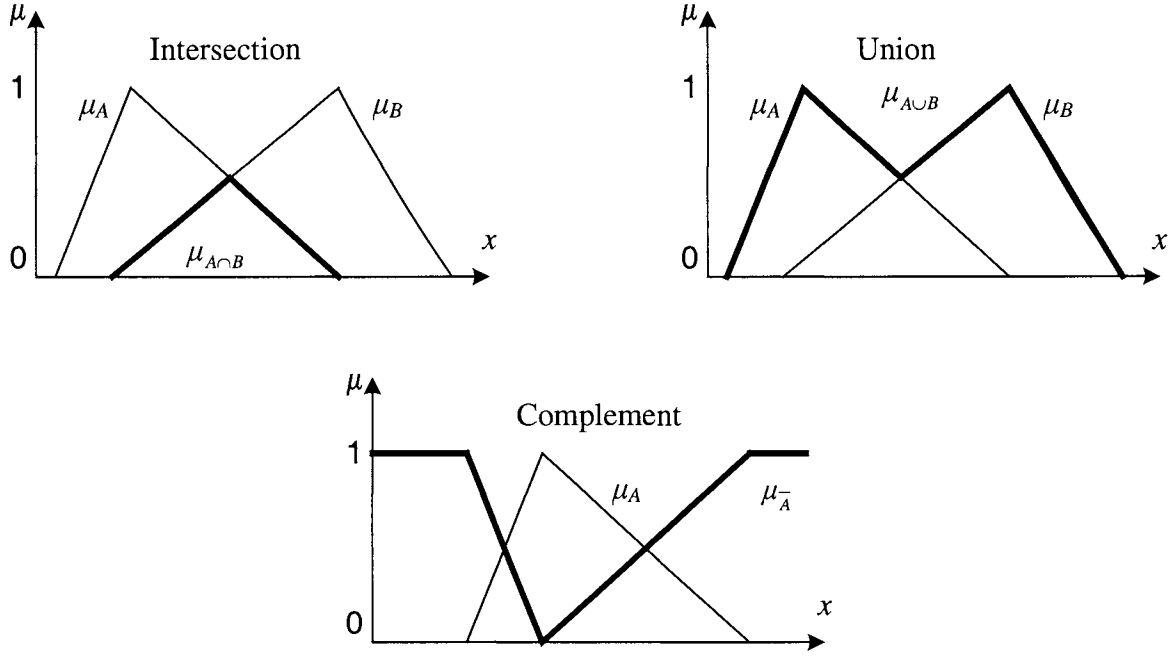


Figure 2.2 Fuzzy logical operations, intersection, union and complement

It can be easily verified that each of the intersection and union operations is:

1. Commutative: $A \cap B = B \cap A$ and $A \cup B = B \cup A$;
2. Associative: $A \cap (B \cap C) = (A \cap B) \cap C$ and $A \cup (B \cup C) = (A \cup B) \cup C$;
3. Idempotent: $A \cap A = A$ and $A \cup A = A$;
4. De Morgan's law: $\overline{A \cap B} = \overline{A} \cup \overline{B}$ and $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

However, these operations do not satisfy the following properties:

5. Law of excluded middle: $A \cup \overline{A} = X$;
6. Non-contradiction principle: $A \cap \overline{A} = 0$.

t-norms and s-norms

Instead of minimum and maximum, many other functions can be used for the logical operations on fuzzy sets. For example, we can also select product ($\mu_A \cdot \mu_B$) and probabilistic sum ($\mu_A + \mu_B - \mu_A \cdot \mu_B$) for intersection and union respectively. Indeed, any function satisfying the properties of t-norms or s-norms (also called t-conorms) is suitable for these two logic operations.

t-norms and s-norms are 2-ary operations mapping $[0, 1]^2$ to $[0, 1]$:

$$t : [0, 1] \times [0, 1] \to [0, 1] \quad \& \quad s : [0, 1] \times [0, 1] \to [0, 1].$$

Properties of t-norms:

1. Commutativity: $t(\mu_A, \mu_B) = t(\mu_B, \mu_A)$;
2. Associativity: $t(\mu_A, t(\mu_B, \mu_C)) = t(t(\mu_A, \mu_B), \mu_C)$;
3. Monotonicity: if $\mu_A \geq \mu_B$ and $\mu_C \geq \mu_D$, then $t(\mu_A, \mu_B) \geq t(\mu_C, \mu_D)$;
4. One identity: $t(\mu_A, 1) = \mu_A$.

Properties of s-norms:

1. Commutativity: $s(\mu_A, \mu_B) = s(\mu_B, \mu_A)$;
2. Associativity: $s(\mu_A, s(\mu_B, \mu_C)) = s(s(\mu_A, \mu_B), \mu_C)$;
3. Monotonicity: if $\mu_A \geq \mu_B$ and $\mu_C \geq \mu_D$, then $s(\mu_A, \mu_B) \geq s(\mu_C, \mu_D)$;
4. Zero identity: $s(\mu_A, 0) = \mu_A$.

Examples of t-norms and s-norms appear in Table 2.2.

Table 2.2 Some widely used t-norms and s-norms

***t*-noms**	***s*-norms**	**Name**
$min\{\mu_A, \mu_B\}$	$max\{\mu_A, \mu_B\}$	minimum / maximum
$\mu_A \bullet \mu_B$	$\mu_A + \mu_B - \mu_A \bullet \mu_B$	product / probabilistic sum
$max\{0, \mu_A + \mu_B - 1\}$	$min\{1, \mu_A + \mu_B\}$	bounded difference / bounded sum
μ_A, if $\mu_B = 1$, μ_B, if $\mu_A = 1$, 0, otherwise.	μ_A, if $\mu_B = 0$, μ_B, if $\mu_A = 0$, 1, otherwise.	drastic product / drastic sum

Fuzzy systems

Structure of fuzzy system

The fuzzy systems, with n ($n \geq 1$) inputs and c ($c \geq 1$) outputs, can be divided into three categories, SISO, MISO and MIMO systems, according to the number of inputs and outputs. A typical fuzzy system consists of four functional blocks such as fuzzification, knowledge base, fuzzy inference and defuzzification [5], as show in Figure 2.3.

The crisp input is fuzzified by the associated input membership function and submitted to fuzzy inference block, which is a decision-making unit and generates fuzzy output through fuzzy reasoning. Defuzzification block calculates crisp output from fuzzy output. Knowledge base, composed of data base and rule base, defines the associated membership function in fuzzification and defuzzification blocks, and provides fuzzy rules to fuzzy inference block.

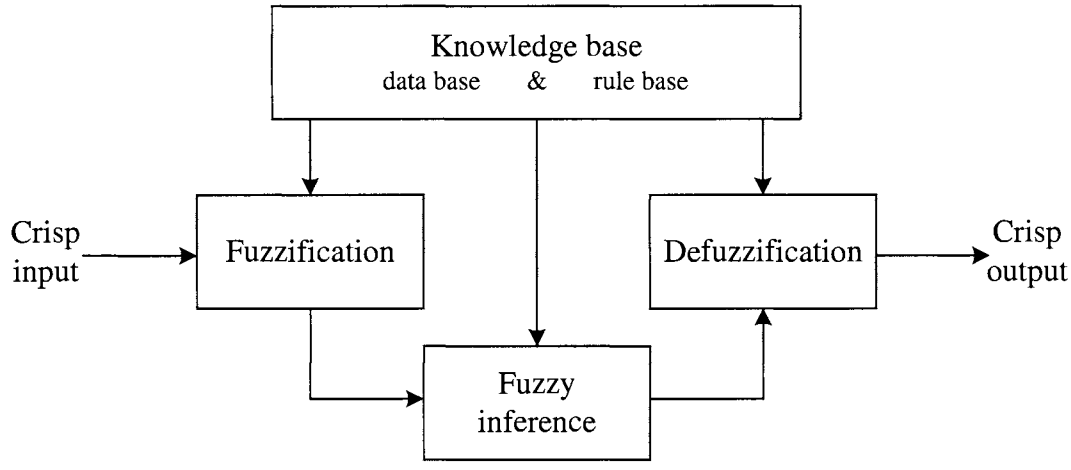


Figure 2.3 Structure of fuzzy system

Fuzzification

The first step of fuzzy inference system is fuzzification, which is to find the degree of matching the input to a set of linguistic variables. The input space of each input variable is divided into different regions, each of which depicts a property of input variable and is associated with a linguistic term as well as a membership function. As shown in Figure 2.4, the range of crisp input variable "temperature" is partitioned into three regions with linguistic terms "cold", "warm" and "hot". Because the support regions of these linguistic sets are overlapped, a crisp input value might be associated with different degrees to different regions at the same time.

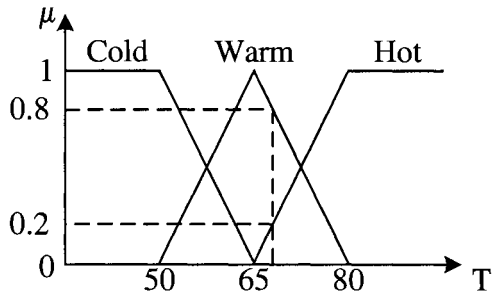


Figure 2.4 Fuzzification of a crisp input

Fuzzy inference

The fuzzy inference is performed using a set of pre-defined fuzzy if-then rules, which can be specified by a human expert or extracted from input-output data pairs and stored in knowledge base. Each rule has an antecedent part and a consequent part and defines the connection between input and output fuzzy variables. An example is "if x_1 is A and x_2 is B,

then y is C". Here, x_1, x_2 and y are called fuzzy variables, while A, B and C are linguistic terms or linguistic variables.

There are two basic models in fuzzy systems, which are differentiated according to their consequent parts [15].

MA (Mamdani-Assilian) [21] *model (logic model):* In MA model, both the input and output are represented by linguistic terms. The antecedent and consequent parts of a rule are typically Boolean expressions of simple clauses. The previous example is of MA model.

TS (Takagi-Sugeno) [30] *model (function model):* In TS model, the antecedent part of a rule is again a Boolean expression of simple clauses, but the consequent part is a function of the input vector $\boldsymbol{x}$. For example, "if x_1 is A and x_2 is B, then $y = a\,x_1 + b\,x_2$".

Fuzzy classifier adopts MA model, however, typically its crisp output is a crisp class label, but not a crisp real value that is often used in other fuzzy systems. The fuzzy output of the fuzzy classifier is a certain membership degree assigned to the output variable label. Consider a fuzzy classifier with n input variable and m output variable labels. The k-th rule of this fuzzy classifier can be expressed as follows:

Rule R_k: **IF** x_1 is $X_{1,k}$ **AND** x_2 is $X_{2,k}$ **AND** ... **AND** x_n is $X_{n,k}$, **THEN** y is Y_k, (2.3.1)

where x_1, x_2 and x_n are n input variables, $X_{1,k}$, $X_{2,k}$ and $X_{n,k}$ are linguistic terms associated with these input variables, Y_k is the class label.

Generally minimum or dot product of membership degrees of all the input variables of a rule is calculated as the firing strength of the rule.

Firing strength: $\alpha_k = \mu_{X1,k} \bullet \mu_{X2,k} \bullet \ldots \bullet \mu_{Xn,k}$ or $min(\mu_{X1,k}, \mu_{X2,k}, \ldots, \mu_{Xn,k})$. (2.3.2)

Inference block makes the decision by performing fuzzy operations on fuzzy values (membership degree values) according to fuzzy rules. The fuzzy values within a fuzzy rule

are aggregated by connective operators such as intersection (AND), union (OR) and complement (NOT).

Defuzzification

Defuzzification is a procedure that calculates the single representative value of a fuzzy set. Several defuzzification strategies have been suggested in [17]. Among them, COA (center of area) and WA (weighted average) are most popular.

The crisp output of COA is the center of gravity of the consequents of fuzzy rules. It is especially suitable for MA model. Assuming output defined on a continuous universe of discourse, COA calculates crisp output according to:

$$y_{COA} = \frac{\int_Y \mu_Y(y)\, y dy}{\int_Y \mu_Y(y) dy}, \tag{2.3.3}$$

where $\mu_Y(y)$ is the aggregated output membership degree.

Figure 2.5 gives an example of COA containing two fuzzy rules and using *minimum* as intersection operation.

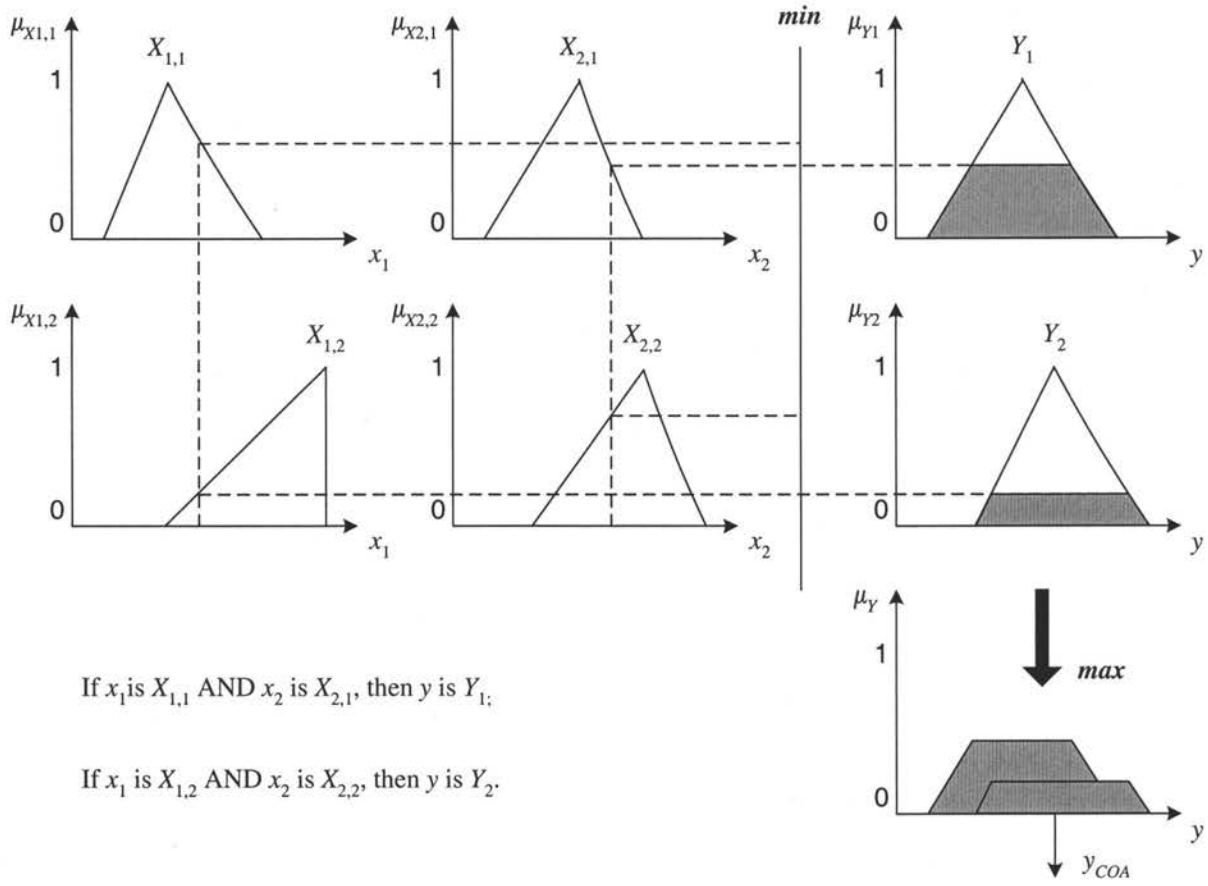


Figure 2.5 An example of COA defuzzification

For TS model with numerical fuzzy output, WA is often used to generate crisp output value according to:

$$y_{WA} = \frac{\sum_{i=1}^{m} \alpha_i y_i}{\sum_{i=1}^{m} \alpha_i}, \tag{2.3.4}$$

where α_i is the firing strength of the i-th rule, m is the number of rules, and y_i is the numerical fuzzy output of the i-th rule.

In fuzzy classifier, defuzzification process is much simpler because its output is a class label. Generally the linguistic term with maximum degree is chosen as its crisp output.

CHAPTER 3. NEURO-FUZZY SYSTEMS

"A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples."

----------Nauck and Kruse [18]

Introduction of neural networks

Artificial neural networks (ANNs) [24] provide a general, practical method for function approximation and pattern classification. By simulating simple nervous systems, they are capable of learning any non-linear continuous function [6] and performing parallel computation. Multi-layer perceptron network (MLP) and radial basis function network (RBF) are two commonly used neural network models.

Generally speaking, MLP, as shown in Figure 3.1, consists of various nodes, which can be divided into three layers such as input layer, hidden layer and output layer. There are directed connections from input to hidden layer then from hidden layer to output layer. Each connection is associated with a weight that can be adjusted by the back-propagation algorithm. It is these adjustable weights that store the knowledge in neural network and endow the learning ability to neural network.

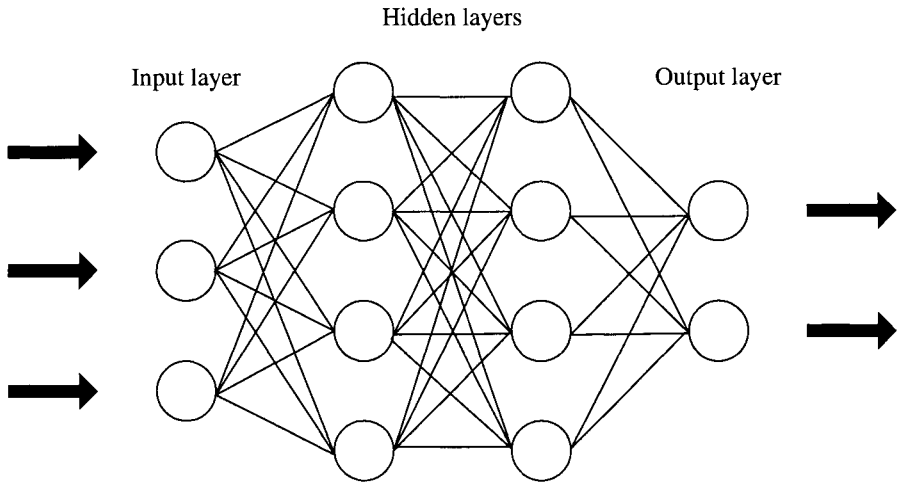


Figure 3.1 Architecture of MLP network

Each node in the hidden and output layers, is a computing unit and contains the input links with weights w_{ij}, an activation function (or transfer function) f, and output links to other nodes as shown in Figure 3.2. Assume k input links are connected to node j, the output o_j of node j is processed by the activation function

$$o_j = f(I_j), \tag{3.1.1}$$

$$I_j = \sum_{i=1}^{k} w_{ij} x_i, \tag{3.1.2}$$

where x_i is the input of node j from node i at the previous layer, and w_{ij} is the weight to link from node i to node j.

Sigmoid function, for its differentiability and continuity, is usually chosen as activation function.

$$f(I) = \frac{1}{1+e^{-aI}}. \tag{3.1.3}$$

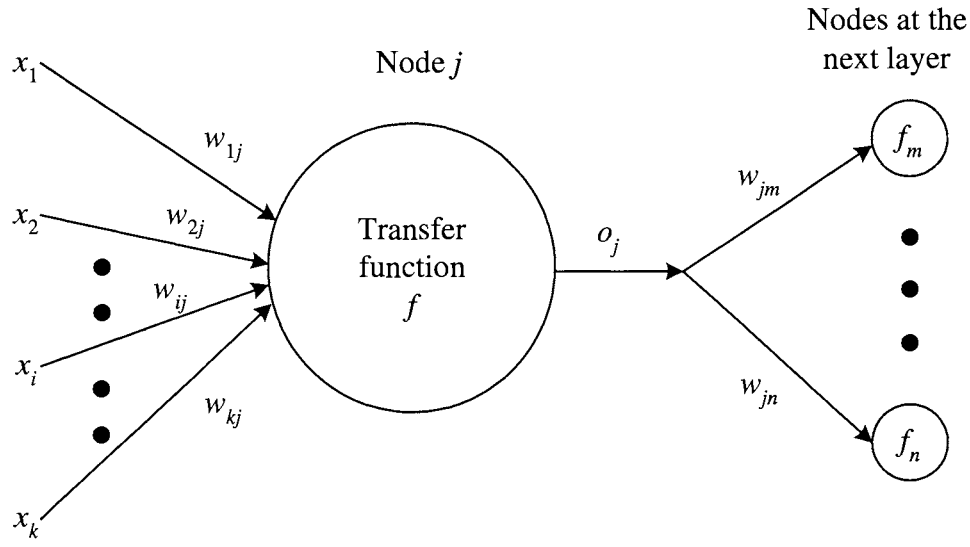


Figure 3.2 Node j with its input/output values in a MLP network

Back-propagation algorithm, which is used to adjust weights in neural networks, brings learning capability to neural networks and thus makes them an attractive method for function approximation and pattern classification. Back-propagation employs gradient descent to attempt to minimize the squared error between the network output values and the target values for these outputs. Let E denote the error function, n is the number of training samples, then weight vector $\boldsymbol{w}$ is adjusted according to

$$E = \frac{1}{2}\sum_{i=1}^{n}(t_i - o_i)^2, \tag{3.1.4}$$

$$\boldsymbol{w}(n+1) \leftarrow \boldsymbol{w}(n) - \eta\frac{\partial E}{\partial \boldsymbol{w}}, \tag{3.1.5}$$

where t_i and o_i are target value and output value of sample I, and η is a positive constant, called the learning rate, to moderate the degree to which weights are changed at each step.

Neuro-fuzzy systems

Introduction

The main advantage of neural networks is their ability to learn from numerical data. However, the knowledge of them is distributed into the whole network as synaptic weights. Therefore, it is hard to associate meaning with the weights. Fuzzy system contains if-then rules, which are linguistic interpretable and easily incorporate a prior knowledge from a human expert. To utilize both advantages within a single framework, various architectures called neuro-fuzzy systems or fuzzy neural networks have been proposed as a hybrid of fuzzy systems and neural networks.

Horikawa, Furuhashi and Uchikawa [6] presented a fuzzy modeling network (FMN) method, which realizes the process of fuzzy inference by the structure of a neural network and expresses parameters of a fuzzy model by the connection weights of neural network. Lin and Lee [20] introduced a neural network based fuzzy logic control and decision system, which adopts MA fuzzy inference model. Two learning algorithms, self-organized learning algorithm and back-propagation learning algorithm, are used successively to locate initial membership functions and adjust their parameters. Wang [32] proposed a fuzzy basis function network (FBFN) and used back-propagation and orthogonal least-squares algorithm for the adjustments of model parameters. Jang [8] also proposed an adaptive network-based fuzzy inference system (ANFIS), which realizes TS fuzzy model and uses back-propagation algorithm and least-squares estimator method to adjust antecedent and consequent parameters respectively.

General architecture

A general architecture of neuro-fuzzy system is shown in Figure 3.3. For the reason of simplicity, only two inputs and two rules are used in the architecture.

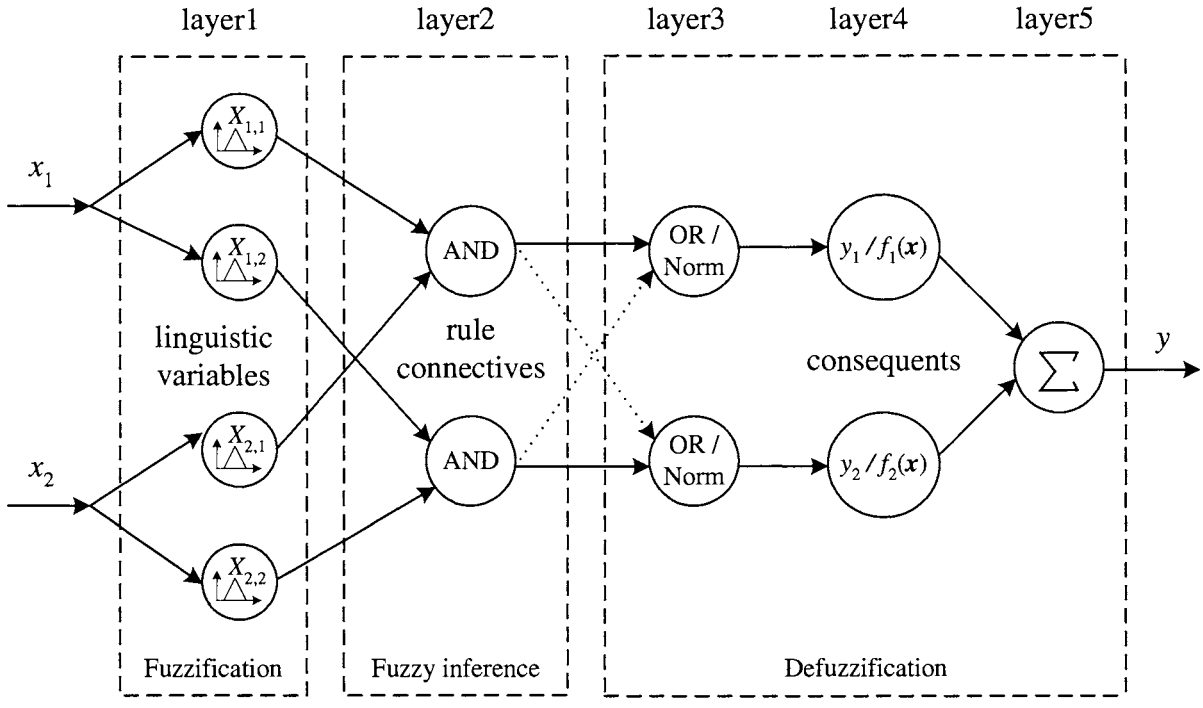


Figure 3.3 General architecture of neuro-fuzzy system

It is a five-layer network, which can implement MA or TS fuzzy inference models. x_1 and x_2 are input variables and y is output variable. The rule base contains only two rules as follows:

Rule1: **IF** x_1 is $X_{1,1}$ **AND** x_2 is $X_{2,1}$, **THEN** y is y_1 (MA model) or $f_1(\boldsymbol{x})$ (TS model). (3.2.1)

Rule2: **IF** x_1 is $X_{1,2}$ **AND** x_2 is $X_{2,2}$, **THEN** y is y_2 (MA model) or $f_2(\boldsymbol{x})$ (TS model). (3.2.2)

where y_i is the center point of the i-th output fuzzy set in MA model, and $f_i(\boldsymbol{x})$ is the output function of the i-th rule in TS model and $\boldsymbol{x}$ is input vector (x_1, x_2).

Layer1 Fuzzification: The crisp inputs x_1 and x_2 are fuzzified by using membership functions of linguistic variables $X_{1,i}$ and $X_{2,i}$. Usually, triangular, trapezoidal or Gaussian function is used. The outputs of layer1 are membership degrees $\mu_{X1,i}$ and $\mu_{X2,i}$.

Layer2 Rule connectives: Each node at this layer corresponds to an if-then rule. It performs logic operation among rule antecedents. Usually, minimum or dot product is used, thus the output of this layer is the firing strength of each rule.

$$\alpha_i = \mu_{X1,i} \bullet \mu_{X2,i} \text{ or } min(\mu_{X1,i}, \mu_{X2,i}). \tag{3.2.3}$$

Layer3 OR / normalization: Some rules may produce the same consequent, which are represented by directed dot lines in Figure 3.3. Therefore, OR operation is performed on the firing strengths of these rules. To reduce the computation of defuzzification process, normalization is also performed between the firing strengths which survive after OR operation. The output of this layer can be looked as the weight to each consequent.

$$w_i = \frac{\alpha_i}{\sum_{k=1}^{m} \alpha_k}. \tag{3.2.4}$$

Layer4 Consequents: In MA model, an output linguistic variable is represented by y_i, which is the center point of the region of the corresponding linguistic variable. In TS model, $f_i(\boldsymbol{x})$ is used, which is the output function of the i-th rule.

Layer5 Summation: The layer computes the overall output as the summation of the incoming signals. In this neuro-fuzzy system architecture, a weighted average defuzzification is performed by nodes from layer3 to layer5.

$$y = \frac{a_i y_i}{\sum_{k=1}^{m} a_k} \quad \text{or} \quad \frac{a_i f_i(x)}{\sum_{k=1}^{m} a_k}. \tag{3.2.5}$$

A neuro-fuzzy classifier adopts MA inference model. However, its output is a crisp class label or a class label with membership degree (unlike the output of a general neuro-fuzzy system that outputs a crisp real value). Therefore, its architecture, shown in Figure 3.4, is a slightly different from that of neuro-fuzzy system.

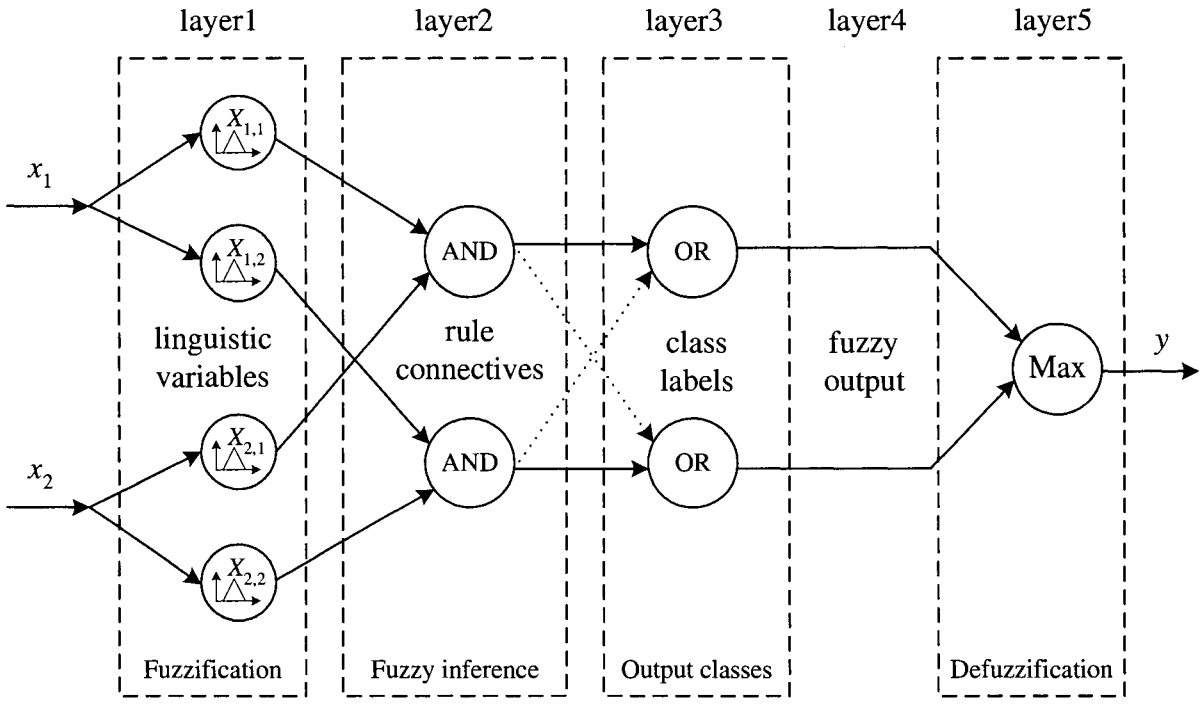


Figure 3.4 General architecture of neuro-fuzzy classifier

In Figure 3.4 layer1 and layer2 remain the same as those in neuro-fuzzy system. However, in layer3 normalization as in (3.2.4) is no longer needed and the output of this layer is the membership degree of the corresponding class. Layer4 is no longer needed and layer5 becomes just a simple maximum selector that outputs the class label with maximum membership degree.

Learning in neuro-fuzzy systems

Generally fuzzy if-then rules come from the knowledge of human expert. However, sometime no human expert is available or even an expert cannot clearly specify the rules he uses. This stimulates interests in fuzzy research community to generate a fuzzy system automatically from the data. Therefore, various learning algorithms, such as back-propagation, least squares, unsupervised clustering and genetic algorithm (GA), have found their application in the field.

Learning problem in neuro-fuzzy system can be divided into two parts, structure identification and parameter identification [28]. The former is related to finding a suitable number of fuzzy rules and a proper partitioning of input space. The latter deals with adjustment of system parameters, such as parameters of membership functions.

The partitioning of input space is one of the important issues in structure identification. The antecedents of fuzzy rules partition the input space into a number of local fuzzy regions, while the consequents describe the behavior within a given region. The partition style can be classified into grid partition, tree partition, and scatter partition, shown in Figure 3.5.

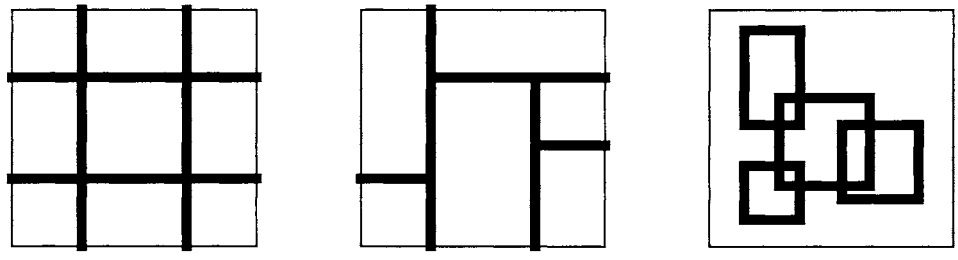


Figure 3.5 Input space partitions

In the grid partition scheme, fuzzy rules share membership functions for each input variables. Wang [33] proposed a general method to produce fuzzy rules by partitioning the input space into equal-width regions. This kind of partitioning is subject to "curse of dimensionality", a problem of exponential explosion in the number of rules as the number of input variables increases. Tree partition can avoid such a problem and the number of partitions implies a corresponding number of rules. Fuzzy Decision tree [10], *k-d* tree partitioning [1], and genetic algorithm [11] are used for this approach. Lin [20] also employed a self-organized learning algorithm to locate initial membership functions and to find the presence of rules. However, the most popular methods are fuzzy clustering, such as fuzzy c-means clustering [2], mountain clustering [35], and subtractive clustering [4]. The clustering methods can also be used to construct scatter partition.

After partitioning of input space, the initial rules are determined. When using fuzzy clustering, each cluster represents a certain region in system input space, and corresponds to a rule in the rule base. The fuzzy sets are obtained by projecting the cluster onto the domain of various input space. The numbers of these rules may not be minimal, so similarity driven simplification [26] and genetic multi-objective optimization [25] can be used to reduce rule base complexity and keep the output accuracy.

After the rules have been found, the entire fuzzy system structure is established and the next phase is to adjust system parameters. To tune antecedent parameters and / or weights of importance, back-propagation algorithm seems to be a dominant choice. This learning approach for the parameters is the main feature that introduces neural network structure into fuzzy system. For the fuzzy system with TS model, the consequent of each rule is a combination of input values and the coefficients in this combination should be estimated

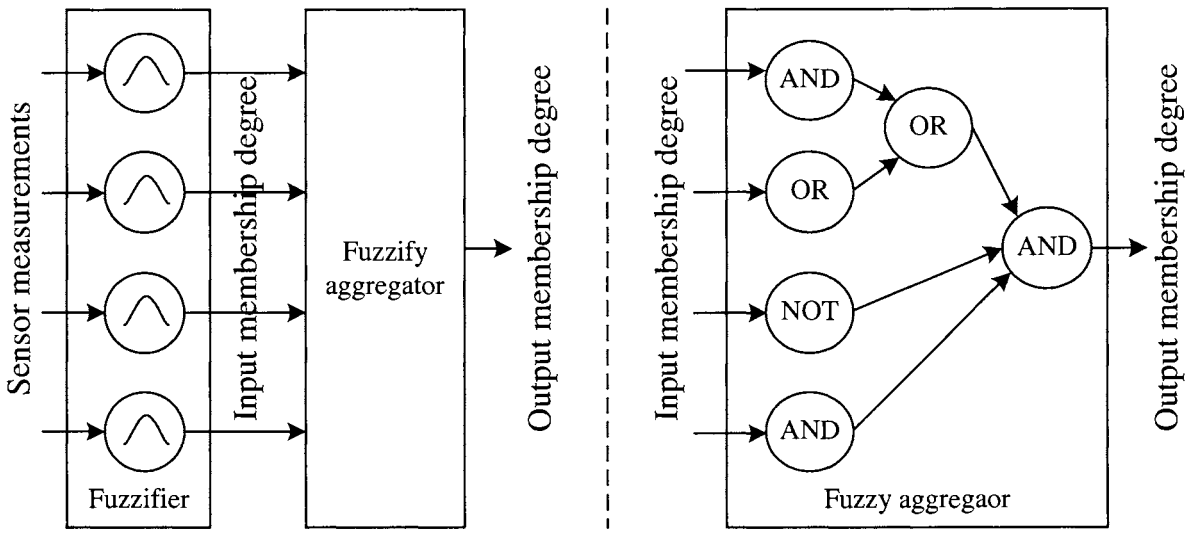
apart from other parameters. To adjust these consequent parameters, least squares algorithm is more widely used.

CINET classifier

Architecture of CINET classifier

CINET (Continuous Inference Network), proposed by Stover and Gibson [29] of Applied Research Laboratory at Pennsylvania State University, is a multi-node fuzzy classification system. Each node in CINET is a classifier that generates a confidence factor for the existence of some property in the sensory data. Each classifier contains various internal nodes as a neural network does.

CINET classifier, shown in Figure 3.6, is a cascade of fuzzifier and fuzzy-aggregator [14]. Fuzzifier is a set of maps, one per input property, which produce membership degrees from crisp sensory data. Then these membership degrees of input properties are submitted to fuzzy aggregator to generate a class label with its membership degree specifying the existence of some interesting output property. Fuzzy aggregator encodes a fuzzy rule and is graphically represented in a topology similar to a neural network. Typical aggregation functions of nodes in fuzzy aggregator are "AND", "OR" and "NOT" of ordinary language. Thus, each CINET classifier performs fuzzy classification based on crisp sensory data and fuzzy logic operations.



(a) CINET classifier (b) Topology of fuzzy aggregator

Figure 3.6 Architecture of CINET classifier

The example CINET classifier depicted in Figure 3.7 simulates a simple inference process of a bat. To capture its prey, a bat sends out sonar to detect the size and speed of the flying object. If the object is small and flying slowly, the bat may think the object is a prey that it can capture. Therefore, the classifier used by the bat contains only one rule, "if the size is small and the speed is slow, then it is a prey".

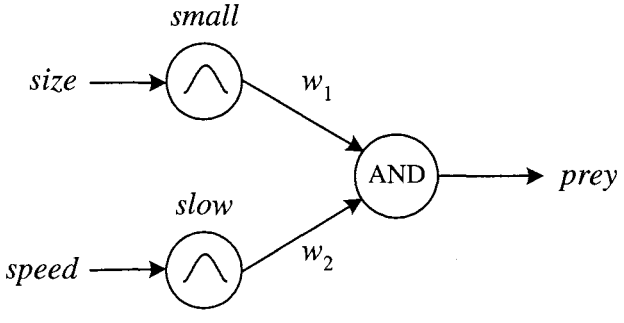


Figure 3.7 CINET classifier for a bat

Fuzzifier

The fuzzifier consists of a set of maps, one per input property, where each map outputs a real value in unit interval. Based on the measurement, each map computes the membership degree to which the input property signifies the desired output property. A sinusoidal-form function, called FUZZIFY function, is used in the CINET for such a computation. FUZZIFY function can be decomposed into two parts, FUZZIFY$^-$ function and FUZZIFY$^+$ function.

FUZZIFY: $R \rightarrow [0, 1]$ is parameterized by four "corner" parameters, $a, b, c, d \in R$ and is defined over $x \in R$:

$$\text{FUZZIFY}^-_{a,b,c,d}(x) := \begin{cases} c & \text{if } x \le a, \\ d & \text{if } x \ge b, \\ sin[\pi(\frac{x-a}{b-a} - 0.5)]\frac{(d-c)}{2} + \frac{(c+d)}{2} & \text{otherwise.} \end{cases} \tag{3.3.1}$$

$$\text{FUZZIFY}^+_{a,b,c,d}(x) := \text{FUZZIFY}^-_{a,b,c,d}(2b - x). \tag{3.3.2}$$

$$\text{FUZZIFY}_{a,b,c,d}(x) := \begin{cases} \text{FUZZIFY}^-_{a,b,c,d}(x) & \text{if } x \le b, \\ \text{FUZZIFY}^+_{a,b,c,d}(x) & \text{otherwise.} \end{cases} \tag{3.3.3}$$

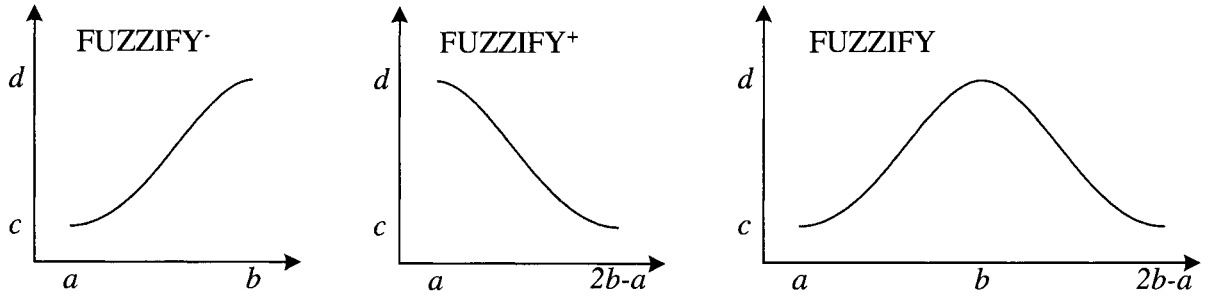


Figure 3.8 FUZZIFY$^-$, FUZZIFY$^+$, and FUZZIFY functions

Fuzzy aggregator

The fuzzy aggregator is a map from a unit hypercube to the unit interval, with a certain canonical form that is composed of the standard connectives of necessity, sufficiency, and complementarity.

Necessity connective: It is called AND, which is used to derive the membership degree of an output property from the knowledge of the membership degrees and significance degrees of its necessity input properties. Significance degree is the weight assigned to each input property to specify their importance with respect to the existence of the output property. A necessity input property means that its non-existence implies the non-existence of the output property.

Given n necessity input properties with membership degrees $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ and their significance degrees $\boldsymbol{w} = [w_1, \ldots, w_n]^T$, their AND is defined as:

$$\mathrm{AND}_{\boldsymbol{w}}(x) = [\prod_{i=1}^{n} (1 - w_i + w_i x_i)]^{0.1+0.9\exp[-0.3(\sum_i w_i - 1)]}. \tag{3.3.4}$$

The output of necessity connective is obtained by first taking the product of the normalized input membership degrees, and next scaling the product by raising it by a certain exponent.

Sufficiency connective: It is called OR, which is used to derive the membership degree of an output property from the knowledge of the membership degrees and significance degrees of its sufficiency input properties. Significance degree has the same meaning as for the necessity connective. A sufficiency input property means that its existence implies the existence of the output property.

Given n sufficiency input properties with membership degrees $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ and their significance degrees $\boldsymbol{w} = [w_1, \ldots, w_n]^T$, their OR is defined as:

$$\mathrm{OR}_{w}(x) = \mathrm{FUZZIFY}^{-}{}_{(2\max_i\{w_i x_i - \sqrt{n}\}),\,(2\max_i\{w_i x_i - 1\}),\,\sqrt{n},\,1}\left(\sqrt{\sum_i (w_i x_i)^2}\right). \tag{3.3.5}$$

The output of sufficiency connective is obtained by first taking the Euclidean norm of the normalized input membership degrees, and next scaling the norm by applying the FUZZIFY^{-} operation.

Complementarity connective: It is called NOT, which is used to derive the membership degree of an output property from the knowledge of the membership degrees and ambiguity degrees of its complementarity input properties. By changing the weight assigned to the input property, ambiguity degree specifies the degree to which our knowledge is incomplete about that input property. A complementarity input property means that its existence implies the non-existence of the output property.

Given a complementarity input property with the membership degree $x \in [0, 1]$, its NOT is defined as:

$$\mathrm{NOT}(x) = \mathrm{AND}_{I}(1 - x, 1 - \alpha), \tag{3.3.6}$$

where $\alpha \in [0, 1]$ is the ambiguity degree of the input measurement. Note that when ambiguity degree is zero, the output is (1-x), which coincides with standard definition of "NOT" for a fuzzy set. However, if ambiguity degree is non-zero, the output is less than (1-x), which means the membership degree gets diminished.

Generally speaking, the output of a CINET classifier is the membership degree of some interesting property, and this output membership degree can also be submitted to other CINET classifiers at the next layer if the further inference is needed. The only difference between these CINET classifiers is that the fuzzifier part is not required in CINET classifier at the next layer, because its input is already a fuzzy value, not a crisp one.

Characteristics of CINET classifier

CINET classifier is a neuro-fuzzy system, because it employs neural network topology to facilitate fuzzy classification. Each CINET classifier is a MISO system whose output is the membership degree of output property. Compared with other neuro-fuzzy systems, CINET classifier has its own characteristics:

1. Only binary partition of input space is needed within a CINET classifier.
2. In aggregator part arbitrary composition of connectives is possible.

In other neuro-fuzzy systems, the space of an input is divided into several regions, each of which is associated with a linguistic term to specify a property of input variable. The antecedent part of a rule is an AND clause involving one linguistic term for each input variable. OR operation is used for multiple rules producing the same output property. So the structure of other neuro-fuzzy systems contains multiple AND nodes at the same layer and only one following OR node at the next layer. Different properties of the same input variable are used in rule base, thus several membership functions are associated with an input variable at the same time.

In CINET classifier, to classify a certain output property we assume only one property of each input variable is required. Different properties of an input variable are only used when multiple outputs are required in an MIMO classifier system. Therefore the rule base of a CINET contains only one rule, in which each input property is used only once and only one membership function is associated with each input variable. These input properties might be aggregated using a logical network of an arbitrary topology.

These differences between the structure of CINET and other neuro-fuzzy systems are due to the methods by which we obtain the model of fuzzy system. Generally other fuzzy systems are automatically learned from data by some learning algorithms, which can use multiple simple rules to achieve high accuracy. CINET models usually come from some human experts with plentiful experience and strong ability of generalization. These experts may form a single, but complicated rule, than various simple ones.

CHAPTER 4. ENHANCEMENTS OF CINET CLASSIFIER

Some enhancements of CINET classifier are proposed. First, we prove that the problem of learning the range of input membership function of CINET classifier is a Linear Mixed Integer Programming (LMIP) problem [7]. Then the necessity and sufficiency functions are simplified to satisfy some algebraic properties. Finally we define ambiguity degree to deal with the randomness in input measurements and its calculation method is also given.

Range of input membership functions

Problem formulation

As introduced above, CINET classifier can implement quite complicated rule, which makes its automatic inference from data difficult. In fact, the logical model of CINET comes from the knowledge of some experts, unlike other fuzzy systems that try to learn their models from the training data automatically. On the other hand, the partitioning of input space becomes much simpler, since each input variable in a CINET classifier is only associated with one membership function, which partitions the corresponding space into two parts. One part, called the range of membership function, is associated with all points whose membership degrees are greater than zero, and specifies the existence of the corresponding input property. Again we assume we have a priori knowledge about which form (FUZZIFY^-, FUZZIFY^+ or FUZZIFY) the membership function takes. Among the four parameters of FUZZIFY function, we assume that c and d are known values (typically 0 and 1 respectively).

Now the question is: given the logical structure and the shape of membership functions of a CINET classifier, how can we determine the range of each input property, namely the two parameters, a and b, from the sample data.

We formulate the above input range determination problem as follows: A CINET classifier, with its structure given, has n continuous input variables $(x_1, \ldots, x_n)$ and one continuous output variable y. The maximum range $[L_i, U_i]$ of the input variable x_i is given, otherwise, we can take the maximum and minimum values from training samples and use them as the extreme values. The output variable, which is unit interval valued, is the membership degree of the corresponding output property. We assign the label Y to the output property for the j-th sample $(x_1^{(j)}, \ldots, x_n^{(j)})$, if $y^{(j)} > 0$, and otherwise, the label is $\overline{Y}$. Given enough number of sample data, we want to determine the interval $[l_i, u_i]$ for each input variable x_i, in which its membership degree is greater than zero. The desired FUZZIFY function is plotted in Figure 4.1. Obviously, $l_i = L_i$ if the desired membership function is FUZZIFY$^-$, and $u_i = U_i$ if FUZZIFY$^+$ desired.

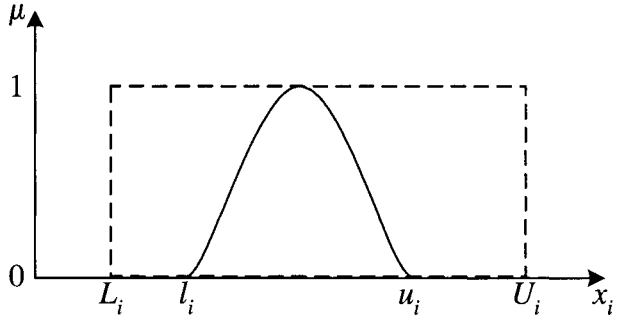


Figure 4.1 Determine the range of membership function

AND classifier

Let us consider one of the simplest classifiers, AND classifier, containing one necessity connective function with two input variables x_1 and x_2. The structure is shown in Figure 4.2 and its corresponding $Rule_{AND}$ is

$$\textit{Rule}_{AND}\text{:} \qquad \textbf{IF } x_1 = X_1 \textbf{ AND } x_2 = X_2, \textbf{ THEN } y = Y, \tag{4.1.1}$$

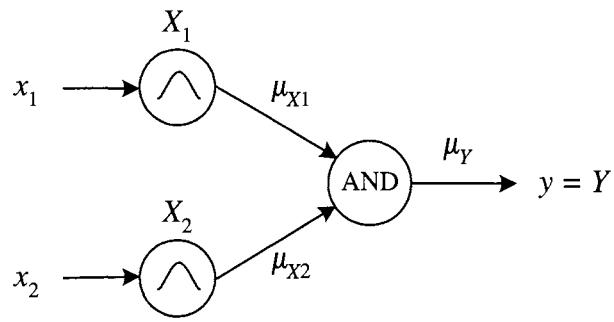


Figure 4.2 AND classifier

where the linguistic terms X_1 and X_2 specify the input properties, Y is the output property, and μ_{X1}, μ_{X2} and μ_Y are the membership degrees of the inputs and the output.

We have m samples, each of which is a 4-dimension vector

$$(x_1^{(j)}, x_2^{(j)}, y^{(j)}, \mu_Y^{(j)}), \text{ where } j = 1, 2 \ldots, m, \text{ and}$$

$$x_1^{(j)} \in [L_1, U_1],\ x_2^{(j)} \in [L_2, U_2],\ y^{(j)} \in \{Y, \overline{Y}\}, \text{ and } \mu_Y^{(j)} \in [0, 1]. \tag{4.1.2}$$

We call the j-th sample a positive sample if $\mu_Y^{(j)} > 0$. Thus, each positive sample is an element of set $\{(x_1, x_2, y, \mu_Y) \mid y = Y\}$ because the output is labeled as Y only when the output membership degree is greater than zero. Let $[l_1, u_1]$ and $[l_2, u_2]$ denote the ranges of membership functions of x_1 and x_2 respectively. According to $Rule_{AND}$, each positive sample satisfies the conjunction of two inequalities $\{l_1 \le x_1 \le u_1$ **AND** $l_2 \le x_2 \le u_2\}$. Therefore, each

positive sample can be regarded as a point in 2-dimension input space and should be bounded in the shadowed region of $\{l_1 \le x_1 \le u_1$ **AND** $l_2 \le x_2 \le u_2\}$ as shown in Figure 4.3.

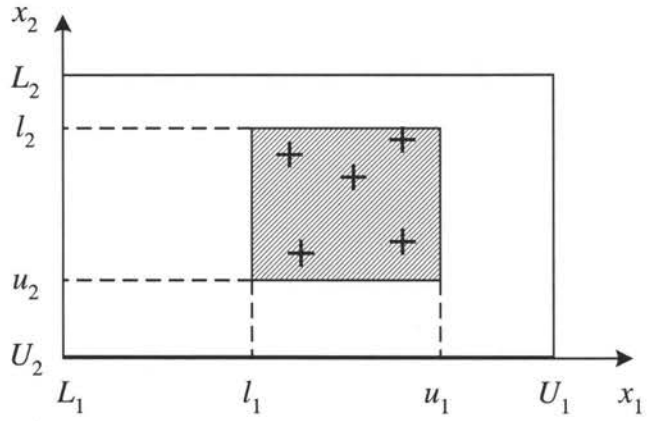


Figure 4.3 Ranges of membership functions in AND classifier

This is an axis parallel rectangle-learning problem [12]: given multiple sample points, learn an unknown axis parallel rectangle consistent with the positive samples. Each positive sample describes a point within the rectangle. According to Probably Approximately Correct (PAC) learning theory [31], a PAC learnable algorithm solving this problem is to draw a minimal rectangle that bounds all positive sample points and use it as a reasonable hypothesis of that target rectangle. This algorithm can make sure that all the points within the hypothesis rectangle are also within the target rectangle, but those outside points may or may not be. Given more and more sample points, the hypothesis rectangle will converge to the target asymptotically.

Following the above idea, we can also draw a minimal rectangle that bounds all positive samples to approximate the range of input membership function. Each point within the rectangle would also be an element of set $\{(x_1, x_2, y, \mu_Y) \mid y = C\}$. Although we cannot

draw any conclusion about those points outside the rectangle, the result is reasonable and acceptable because we have no knowledge about those unseen outside points. Therefore, our solution is to choose the minimal range that bounds all positive sample points for each input membership function. Figure 4.4 depicts such minimal ranges in AND classifier.

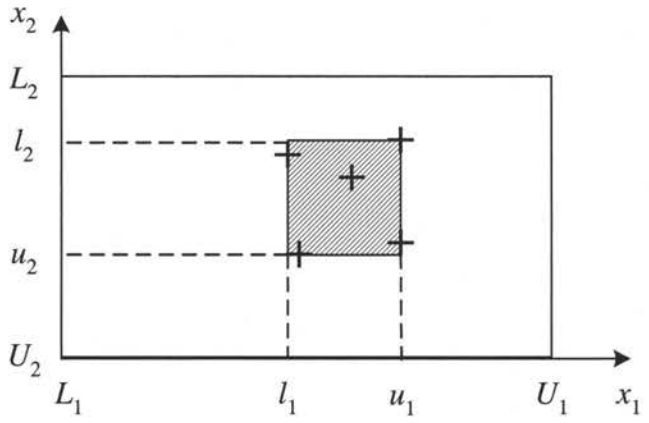


Figure 4.4 Minimal ranges of membership functions in AND classifier

We expect to minimize both the ranges (u_1 - l_1) and (u_2 - l_2) at the same time. The adjustments of (u_1 - l_1) and (u_2 - l_2) are independent, which means increasing or decreasing the range of one input membership function would not affect that of the other, so we can obtain the minimal ranges separately or together. That is, $min(u_1 - l_1)$ and $min(u_2 - l_2)$ is equivalent to $min[(u_1 - l_1)(u_2 - l_2)]$, or $min[(u_1 - l_1) + (u_2 - l_2)]$. Since different inputs may posses different unit of measurements, we choose to minimize the normalized range. That is

$$min(\frac{u_1 - l_1}{U_1 - L_1}) \quad \text{and} \quad min(\frac{u_2 - l_2}{U_2 - L_2}), \tag{4.1.3}$$

$$\text{or} \qquad min\left[(\frac{u_1 - l_1}{U_1 - L_1})(\frac{u_2 - l_2}{U_2 - L_2})\right], \tag{4.1.4}$$

$$\text{or} \qquad min\left[(\frac{u_1 - l_1}{U_1 - L_1}) + (\frac{u_2 - l_2}{U_2 - L_2})\right]. \tag{4.1.5}$$

For simplicity we prefer (4.1.5) to (4.1.4) although either is acceptable.

OR classifier

Let us consider another simple classifier, OR classifier, containing one sufficiency connective function with two input variables x_1 and x_2. The structure is shown in Figure 4.5 and its corresponding $Rule_{OR}$ is:

***Rule*$_{OR}$:** **IF** $x_1 = X_1$ **OR** $x_2 = X_2$, **THEN** $y = Y$, (4.1.6)

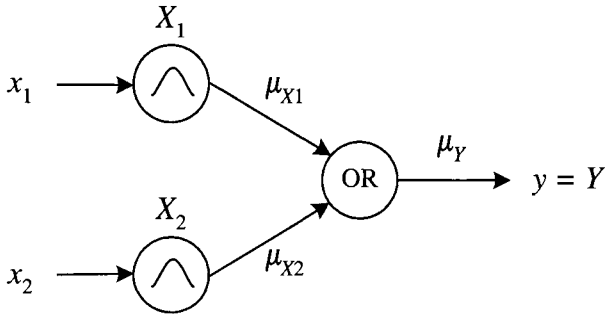


Figure 4.5 OR classifier

where the linguistic terms X_1 and X_2 specify the input properties, Y is the output property, and μ_{X1}, μ_{X2} and μ_Y are the membership degrees of the inputs and the output.

We have m samples, each of which is a 4-dimension vector

$$(x_1^{(j)}, x_2^{(j)}, y^{(j)}, \mu_Y{}^{(j)}), \text{ where } j = 1,2 \ldots, m, \text{ and}$$

$$x_1^{(j)} \in [L_1, U_1],\ x_2^{(j)} \in [L_2, U_2],\ y^{(j)} \in \{Y, \bar{Y}\}, \text{ and } \mu_Y{}^{(j)} \in [0, 1]. \tag{4.1.7}$$

We call the j-th sample a positive sample if $\mu_Y{}^{(j)} > 0$. Thus, each positive sample is an element of set $\{(x_1, x_2, y, \mu_Y) \mid y = Y\}$ because the output is labeled as Y only when the output membership degree is greater than zero. Let $[l_1, u_1]$ and $[l_2, u_2]$ denote the ranges of membership functions of x_1 and x_2 respectively. According to $Rule_{OR}$, each positive sample satisfies the disjunction of two inequalities $\{l_1 \le x_1 \le u_1$ **OR** $l_2 \le x_2 \le u_2\}$. Therefore, each positive sample can be regarded as a point in 2-dimension input space and should be bounded in the shadowed region $\{l_1 \le x_1 \le u_1$ **OR** $l_2 \le x_2 \le u_2\}$ as shown in Figure 4.6.

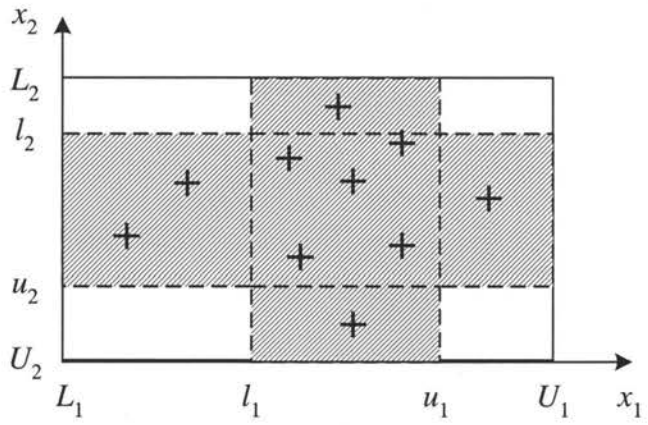


Figure 4.6 Ranges of membership functions in OR classifier

We expect to minimize both the ranges $(u_1 - l_1)$ and $(u_2 - l_2)$ simultaneously, however, this may not be possible. Figure 4.7 shows the reason. To bound all positive points in the shadowed region $\{l_1 \le x_1 \le u_1$ **OR** $l_2 \le x_2 \le u_2\}$, that is, as small as possible, we have two choices that either increase $(u_1 - l_1)$ and decrease $(u_2 - l_2)$ or decrease $(u_1 - l_1)$ and increase $(u_2 - l_2)$. Therefore, the adjustments of $(u_1 - l_1)$ and $(u_2 - l_2)$ are not independent. We have to deal with this tradeoff to balance our requirement of minimizing both the ranges of two input membership functions.

Thus a simple way is to choose $min\{(u_1 - l_1) + (u_2 - l_2)\}$ as the desired minimum function. As before, we consider normalized ranges, that is,

$$min\left[(\frac{u_1 - l_1}{U_1 - L_1}) + (\frac{u_2 - l_2}{U_2 - L_2})\right]. \tag{4.1.8}$$

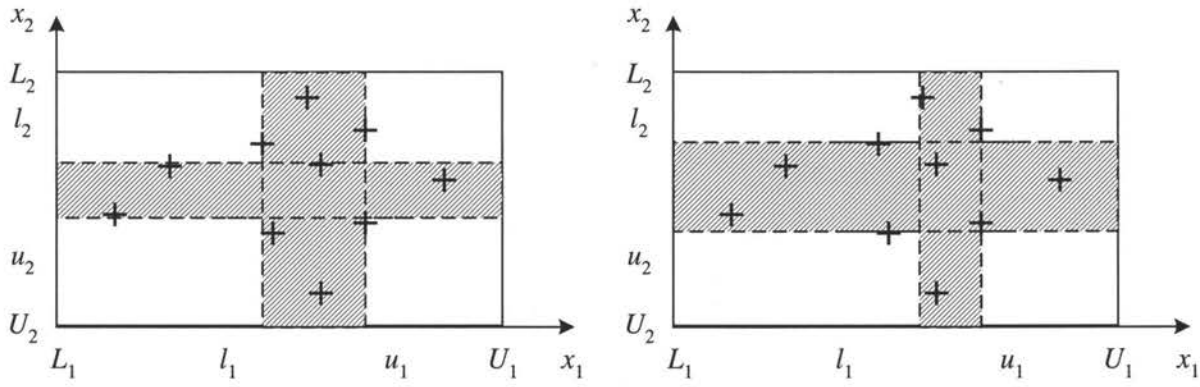


Figure 4.7 Two possible minimizations in OR classifier

Linear Mixed Integer Programming (LMIP) formulation

Based on the above discussion about AND and OR classifiers, we can deal with the CINET classifier which can have an arbitrary topology and represent a complicated rule involving AND, OR and NOT operations.

In CINET classifier, each input variable is only associated with one membership function, which represents a fuzzy set. If we are only interested in knowing the existence of an output property (i.e., positive output membership degree), then it suffice to view the output as well as each input as a crisp set, where the membership degree in the crisp set is affirmative if and only if its fuzzy membership degree is positive. For example, the fuzzy set $\{X_1\}$ for input variable x_1, represents the set $\{l_1 \leq x_1 \leq u_1\}$, where $\mu_{X1}(x_1) > 0$. For such

purposes, we can apply the distributivity law of propositional logic [22] to any rule in CINET classifier as follows:

$$(x_1 = X_1 \textbf{ OR } x_2 = X_2) \textbf{ AND } (x_3 = X_3)$$

$$= (x_1 = X_1 \textbf{ AND } x_3 = X_3) \textbf{ OR } (x_2 = X_2 \textbf{ AND } x_3 = X_3), \tag{4.1.9}$$

equivalently,

$$(l_1 \le x_1 \le u_1 \textbf{ OR } l_2 \le x_2 \le u_2) \textbf{ AND } (l_3 \le x_3 \le u_3)$$

$$= (l_1 \le x_1 \le u_1 \textbf{ AND } l_3 \le x_3 \le u_3) \textbf{ OR } (l_2 \le x_2 \le u_2 \textbf{ AND } l_3 \le x_3 \le u_3). \tag{4.1.10}$$

Thus, applying (4.1.9) to the rule represented by a CINET classifier, we could always transform the rule ($Rule_{CINET}$) to a disjunction of multiple conjunction sub-clauses as follows:

$$\begin{aligned} \textbf{\textit{Rule}}_{CINET}\textbf{:} \quad & \textbf{IF } (x_{1,1} = X_{1,1} \textbf{ AND } x_{1,2} = X_{1,2} \textbf{ AND } \ldots \textbf{ AND } x_{1,k1} = X_{1,k1}) \\ & \ldots \\ & \textbf{OR } (x_{i,1} = X_{i,1} \textbf{ AND } x_{i,2} = X_{i,2} \textbf{ AND } \ldots \textbf{ AND } x_{i,ki} = X_{i,ki}) \\ & \ldots \\ & \textbf{OR } (x_{m,1} = X_{m,1} \textbf{ AND } x_{m,2} = X_{m,2} \textbf{ AND } \ldots \textbf{ AND } x_{m,km} = X_{m,km}) \\ & \textbf{THEN } y = Y, \end{aligned} \tag{4.1.11}$$

where m is the number of conjunction sub-clauses, $x_{i,j}$ and $X_{i,j}$ are the j-th input variable and its input property of the i-th sub-clause. $i \in [1, k_i]$ and $k_i \in [1, n]$ where k_i is the length of the i-th sub-clause and n is total number of input variables. Note that within the i-th sub-clause, $x_{i,j}$ and $X_{i,j}$ are distinct, but between the different sub-clauses they are not necessary to be distinct.

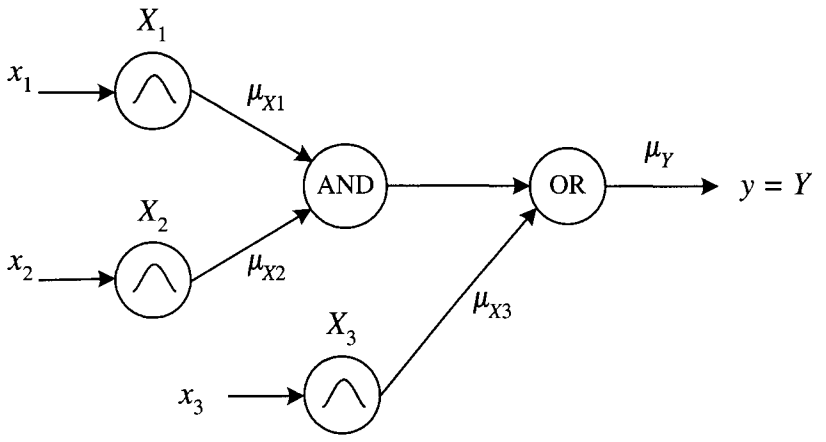


Figure 4.8 Example of a CINET classifier

Consider a CINET classifier as shown in Figure 4.8, it corresponds to the rule

$$\textbf{IF } (x_1 = X_1 \textbf{ AND } x_2 = X_2) \textbf{ OR } (x_3 = X_3), \textbf{ THEN } y = Y, \tag{4.1.12}$$

equivalently,

$$\textbf{IF } (l_1 \le x_1 \le u_1 \textbf{ AND } l_2 \le x_2 \le u_2) \textbf{ OR } (l_3 \le x_3 \le u_3), \textbf{ THEN } y = Y. \tag{4.1.13}$$

We can approximate the ranges by minimizing the summation of all normalized ranges. As to the example CINET classifier shown in Figure 4.8, based on all positive samples we want to determine the boundary points l_1, u_1, l_2, u_2, l_3 and u_3 to minimize the summation of their normalized intervals:

$$\underset{u_1, l_1, u_2, l_2, u_3, l_3}{argmin} \quad \left[\left(\frac{u_1 - l_1}{U_1 - L_1}\right) + \left(\frac{u_2 - l_2}{U_2 - L_2}\right) + \left(\frac{u_3 - l_3}{U_3 - L_3}\right) \right], \tag{4.1.14}$$

subject to the union of following constrains:

$$(l_1 \le x_1^{(i)} \le u_1 \textbf{ AND } l_2 \le x_2^{(i)} \le u_2 \textbf{ AND } L_3 \le x_3^{(i)} \le U_3), \tag{4.1.15}$$

$$\textbf{OR} \quad (L_1 \le x_1^{(i)} \le U_1 \textbf{ AND } L_2 \le x_2^{(i)} \le U_2 \textbf{ AND } l_3 \le x_3^{(i)} \le u_3), \tag{4.1.16}$$

where $x_j^{(i)}$ is the i-th positive sample of input variable j. The length of each constraint equals to the number of input variables and each constraint is an extension of a conjunction sub-clause in (4.1.13).

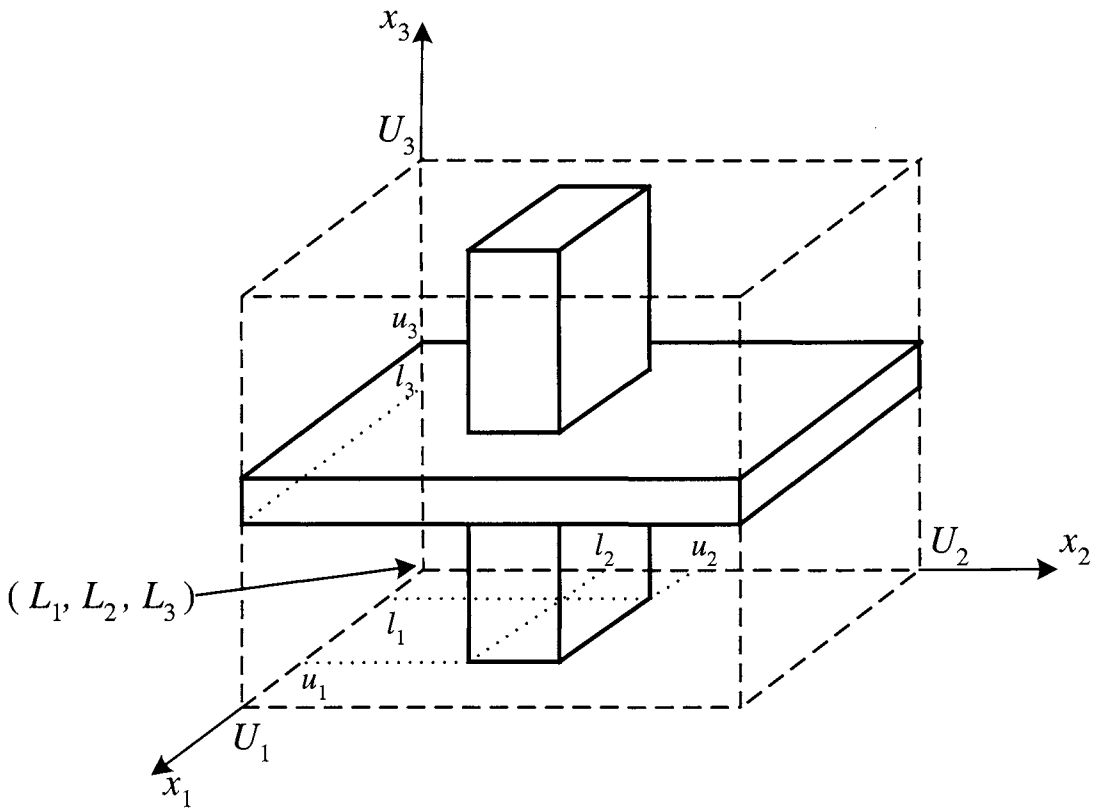


Figure 4.9 Range of membership functions of the example CINET classifier

Figure 4.9 depicts the solution space of (4.1.15) and (4.1.16), which is the union of two cubes. Each cube corresponds to the solution space of a constraint. Using binary valued variables z_i, we can combine these two constraints together to obtain:

$$\underset{u_1, l_1, u_2, l_2, u_3, l_3}{argmin} \quad \left[\left(\frac{u_1 - l_1}{U_1 - L_1}\right) + \left(\frac{u_2 - l_2}{U_2 - L_2}\right) + \left(\frac{u_3 - l_3}{U_3 - L_3}\right) \right], \tag{4.1.17}$$

subject to the constrains:

$$\sum_{i=1}^{m}(z_i l_k^{(i)}) \le x_k \le \sum_{i=1}^{m}(z_i u_k^{(i)}), \tag{4.1.18}$$

$$\sum_{i=1}^{m} z_i = 1, \quad z_i \in \{0,1\}, \tag{4.1.19}$$

$$l_k^{(i)} \in \{l_k, L_k\},\ u_k^{(i)} \in \{u_k, U_k\}, \tag{4.1.20}$$

where $m = 2$, which is the number of constraints, and $k \in \{1, 2, 3\}$. (4.1.19) means that each positive sample point should satisfy one constraint within the union of constraints.

In the problem depicted by (4.1.17) to (4.1.20), $l_k^{(i)}$ and $u_k^{(i)}$ are real values, while z_i are binary variables. So it is a Linear Mixed Integer Programming (LMIP) problem, in which integers z_i are mixed with real values $l_k^{(i)}$ and $u_k^{(i)}$. This problem can be solved by LMIP software such as ILOG CPLEX [7].

We can apply the above LMIP formulation to a general CINET classifier with n input variables and m conjunction sub-clauses described by (4.1.11).

First, we extend each conjunction sub-clause to a full-length constraint with some input variables bounded by extreme points that are fixed and other variables taking values in variable input ranges. The i-th conjunction sub-clause

$$(x_{i,1} = X_{i,1} \textbf{ AND } x_{i,2} = X_{i,2} \textbf{ AND } \ldots \textbf{ AND } x_{i,ki} = X_{i,ki}), \tag{4.1.21}$$

can be extended as the following full-length constraint

$$(x_{i,1} = X_{i,1} \textbf{ AND } \ldots \textbf{ AND } x_{i,ki} = X_{i,ki} \textbf{ AND } x_{i,ki+1} = X_{i,ki+1} \textbf{ AND } \ldots \textbf{ AND } x_{i,n} = X_{i,n}), \tag{4.1.22}$$

which is equivalent to

$$(l_{i,1} \le x_{i,1} \le u_{i,1} \textbf{ AND } \ldots \textbf{ AND } l_{i,ki} \le x_{i,ki} \le u_{i,ki} \textbf{ AND}$$

$$L_{i,ki+1} \le x_{i,ki+1} \le U_{i,ki+1} \textbf{ AND } \ldots \textbf{ AND } L_{i,n} \le x_{i,n} \le U_{i,n}). \tag{4.1.23}$$

Similarly, all the sub-clauses in $\text{Rule}_{\text{CINET}}$ can be expressed as follows:

$$(l_{1,1} \le x_{1,1} \le u_{1,1} \textbf{ AND } \ldots \textbf{ AND } l_{1,k1} \le x_{1,k1} \le u_{1,k1} \textbf{ AND}$$

$$L_{1,k1+1} \le x_{1,k1+1} \le U_{1,k1+1} \textbf{ AND } \ldots \textbf{ AND } L_{1,n} \le x_{1,n} \le U_{1,n})$$

$$\ldots$$

$$\textbf{OR} \quad (l_{i,1} \le x_{i,1} \le u_{i,1} \textbf{ AND } \ldots \textbf{ AND } l_{i,ki} \le x_{i,ki} \le u_{i,ki} \textbf{ AND}$$

$$L_{i,ki+1} \le x_{i,ki+1} \le U_{i,ki+1} \textbf{ AND } \ldots \textbf{ AND } L_{i,n} \le x_{i,n} \le U_{i,n})$$

$$\ldots$$

$$\textbf{OR} \quad (l_{m,1} \le x_{m,1} \le u_{m,1} \textbf{ AND } \ldots \textbf{ AND } l_{m,km} \le x_{m,km} \le u_{m,km} \textbf{ AND}$$

$$L_{m,km+1} \le x_{m,km+1} \le U_{m,km+1} \textbf{ AND } \ldots \textbf{ AND } L_{m,n} \le x_{m,n} \le U_{m,n}). \qquad (4.1.24)$$

From (4.1.24) we can clearly see that the input variables existing in the original sub-clauses are bounded by their extreme points, while the newly added input variables can take any values in their ranges.

Each constraint represents a hypercube in input space and all of them together form a union of these hyper-cubes. By introducing the real valued variables $l_k^{(i)}$ and $u_k^{(i)}$, we can express each constraint with a shorter form. So, (4.1.23) can be rewritten as follows:

$$l_k^{(i)} \le x_k \le u_k^{(i)}, \quad \text{where } k = 1, 2, ..., n, \text{ and} \qquad (4.1.25)$$

$$l_k^{(i)} \in \{l_k, L_k\},\ u_k^{(i)} \in \{u_k, U_k\}, \qquad (4.1.26)$$

where $l_k^{(i)}$ and $u_k^{(i)}$ determine the range of the k-th input variable in i-th constraint.

Further introducing the binary valued variables z_i, we can combine all the disjunctive constraints together and rewrite them as a single constraint. (4.1.24) can be rewritten as follows:

$$\sum_{i=1}^{m} (z_i l_k^{(i)}) \le x_k \le \sum_{i=1}^{m} (z_i u_k^{(i)}), \qquad (4.1.27)$$

$$\sum_{i=1}^{m} z_i = 1, \qquad z_i \in \{0,1\}, \tag{4.1.28}$$

$$l_k^{(i)} \in \{l_k, L_k\},\ u_k^{(i)} \in \{u_k, U_k\}, \tag{4.1.29}$$

where z_i is a binary coefficient of the i-th constraint. (4.1.28) means for each sample point only one constraint is satisfied. Indeed, a sample point should satisfy at least one constraint and can satisfy multiple constraints at the same time, but only one satisfaction is needed and sufficient. $l_k^{(i)}$ and $u_k^{(i)}$ have the same meanings as in (4.1.26).

Our goal is to find the range of each input variable so as to minimize the normalized summation of the intervals they form. That is, based on the constraints from (4.1.27) to (4.1.29), we want to find

$$\underset{u_1, l_1, \cdots, u_n, l_n}{argmin} \quad \left[\left(\frac{u_1 - l_1}{U_1 - L_1}\right) + \cdots + \left(\frac{u_n - l_3}{U_n - L_n}\right) \right]. \tag{4.1.30}$$

This as we establish is a LMIP problem, because $l_k^{(i)}$ and $u_k^{(i)}$ are real values, while z_i are binary integers, and can be solved by ILOG CPLEX [7].

Simplified connective functions

The functions defining the connectives in the fuzzy-aggregator may be enhanced in several ways:

1. Given an input property, its membership degree should be normalized with respect to its significance degree so that the result is an equivalent membership degree with unity significance degree. This will simplify the definition of each connective by defining it for input properties with unity significance degree only.
2. The mathematical definition of the connectives must be such that they satisfy certain basic algebraic properties as described below. Moreover, their definition may be simplified.
3. The definition of a connective must be extended to compute not only the membership degree of the output property from the membership degrees of input properties, but also the ambiguity degree of the output property from the ambiguity degree of the input properties. (discussed in the next section **Ambiguity degree**)

Necessity connective

Consider a necessity input property with membership degree $x \in [0, 1]$ and significance degree $w \in [0, 1]$. We use $v(x, w) \in [0, 1]$ to denote the equivalent membership degree with unity significance degree. Membership degree should be preserved when significance degree is unity, and zero significance degree implies the property must be ignored while computing the AND with another necessity property. So $v(\bullet, \bullet)$ must satisfy the following boundary conditions:

$$v(x, 1) = x. \tag{4.2.1}$$

$$v(x, 0) = 1. \tag{4.2.2}$$

It is easy to verify that $v(\bullet, \bullet)$ cannot be a constant or an affine. Suppose for contradiction, $v(x, w) = ax + bw + c$. Then from (4.2.1) and (4.2.1) we obtain

$$x = v(x, 1) = ax + bw + c \Rightarrow a = 1, \quad \text{and} \tag{4.2.3}$$

$$1 = v(x, 0) = ax + c \Rightarrow a = 0. \tag{4.2.4}$$

Thus we arrive at a contradiction. So we consider the next simplest possible form, the bilinear form, i.e. $v(x, w) = ax + bw + cxw + d$. Then from (4.2.2) and (4.2.1) we obtain

$$1 = v(x, 0) = ax + d \Rightarrow a = 0 \ \& \ d = 1, \quad \text{and} \tag{4.2.5}$$

$$x = v(x, 1) = ax + b + cx + d = b + cx + 1 \Rightarrow b = -1 \ \& \ c = 1. \tag{4.2.6}$$

So we obtain

$$v(x, w) = -w + wx + 1. \tag{4.2.7}$$

Since $-w + wx + 1 = x + (1-w)(1-x)$, it also follows that $v(x, w) \leq 1$, as desired. Incidentally, the formula $v(x, w) = -w + wx + 1$ derived above is the same as that used by Stover [29]. Our derivation provides a justification to the usage of the formula in [29].

Now it suffices to consider necessity connective for input properties with unity significance degree only. The necessity connective must satisfy the following properties:

1. Commutativity: $\text{AND}(x, y) \leq \text{AND}(y, x)$.
2. Associativity: $\text{AND}(\text{AND}(x, y), z) = \text{AND}(x, \text{AND}(y, z))$.
3. Monotonicity: $y \leq z \Rightarrow \text{AND}(x, y) \leq \text{AND}(x, z)$.
4. Boundary condition: $\text{AND}(x, 1) = x$; $\text{AND}(x, 0) = 0$.
5. Continuity in each argument.

It follows from these conditions that $\mathrm{AND}(x, y) \le min(x, y)$. To see this let us suppose, without loss of generality, $x \le y$. Then

$$\mathrm{AND}(x, y) \le \mathrm{AND}(x, 1) = x = min(x, y). \tag{4.2.8}$$

The definition of AND given in [29] satisfies the commutativity, monotonicity, and continuity. We propose the following simpler definition satisfying all the above properties.

$$\mathrm{AND}(x, y) = \frac{xy}{1-(1-x)(1-y)} = \frac{xy}{x+y-xy}. \tag{4.2.9}$$

This definition has a simpler form, moreover, it is easily generalized to multiple input properties by using the associativity property. Figure 4.10 shows the comparison between the necessity function ($\mathrm{AND}_{\mathrm{original}}$) used by Stover in [29] and the new one ($\mathrm{AND}_{\mathrm{modified}}$) according to our definition in (4.2.9). Their difference is small so it is reasonable for us to expect similar results from these two definitions of necessity functions.

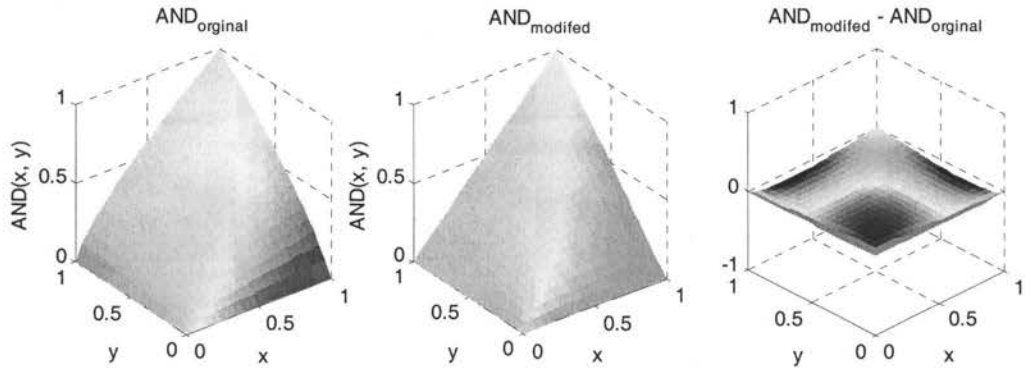


Figure 4.10 Comparison between original and modified necessity functions

Sufficiency connective

Consider a sufficiency input property with membership degree $x \in [0, 1]$ and significance degree $w \in [0, 1]$. We use $v(x, w) \in [0, 1]$ to denote the equivalent membership degree with unity significance degree. Membership degree should be preserved when significance degree is unity, and zero significance degree implies the property must be ignored while computing the OR with another necessity property. So $v(\bullet, \bullet)$ must satisfy the following boundary conditions:

$$v(x, 1) = x. \tag{4.2.10}$$

$$v(x, 0) = 0. \tag{4.2.11}$$

It is easy to verify that $v(\bullet, \bullet)$ cannot be a constant or an affine. So we consider the next simplest possible form, the bilinear form, i.e. $v(x, w) = ax + bw + cxw + d$. Then from (4.2.11) and (4.2.10) we obtain

$$0 = v(x, 0) = ax + d \Rightarrow a = 0 \;\&\; d = 0, \quad \text{and} \tag{4.2.12}$$

$$x = v(x, 1) = ax + b + cx + d = b + cx \Rightarrow b = 0 \;\&\; c = 1. \tag{4.2.13}$$

So we obtain

$$v(x, w) = wx. \tag{4.2.14}$$

It is obvious that $v(x, w) \leq 1$, as desired. Incidentally, the formula $v(x, w) = wx$ derived above is the same as that used by Stover [29]. Our derivation provides a justification to the usage of the formula in [29].

Now it suffices to consider sufficiency connective for input properties with unity significance degree only, and again for simplicity we consider the aggregation of two input properties only. The sufficiency connective must satisfy the following properties:

1. Commutativity: $\mathrm{OR}(x, y) \leq \mathrm{OR}(y, x)$.
2. Associativity: $\mathrm{OR}(\mathrm{OR}(x, y), z) = \mathrm{OR}(x, \mathrm{OR}(y, z))$.
3. Monotonicity: $y \leq z \Rightarrow \mathrm{OR}(x, y) \leq \mathrm{OR}(x, z)$.
4. Boundary condition: $\mathrm{OR}(x, 1) = 1; \mathrm{OR}(x, 0) = x$.
5. Continuity in each argument.

It follows from these conditions that $\mathrm{OR}(x, y) \geq max(x, y)$. To see this let us suppose, without loss of generality, $x \geq y$. Then

$$\mathrm{OR}(x, y) \geq \mathrm{OR}(x, 0) = x = max(x, y). \tag{4.2.15}$$

The definition of OR given in [29] satisfies the commutativity, monotonicity, and continuity. We propose the following simpler definition satisfying all the above properties.

$$\mathrm{OR}(x, y) = \mathrm{NOT}(\mathrm{AND}(\mathrm{NOT}(x), \mathrm{NOT}(y))). \tag{4.2.16}$$

Note that in special case when $\mathrm{NOT}(x) = 1 - x$, the above definition reduces to

$$\mathrm{OR}(x, y) = 1 - \mathrm{AND}(1 - x, 1 - y) = 1 - \frac{(1 - x)(1 - y)}{1 - xy} = \frac{x + y - 2xy}{1 - xy}. \tag{4.2.17}$$

This definition is simpler and also easily generalized to multiple input properties by using the associativity property. Figure 4.12 shows the comparison between the sufficiency function ($\mathrm{OR}_{\mathrm{original}}$) used by Stover in [29] and the new one ($\mathrm{OR}_{\mathrm{modified}}$) according to our definition in (4.2.17). Their difference is small so it is reasonable for us to expect similar results from both of these definitions of sufficiency functions.

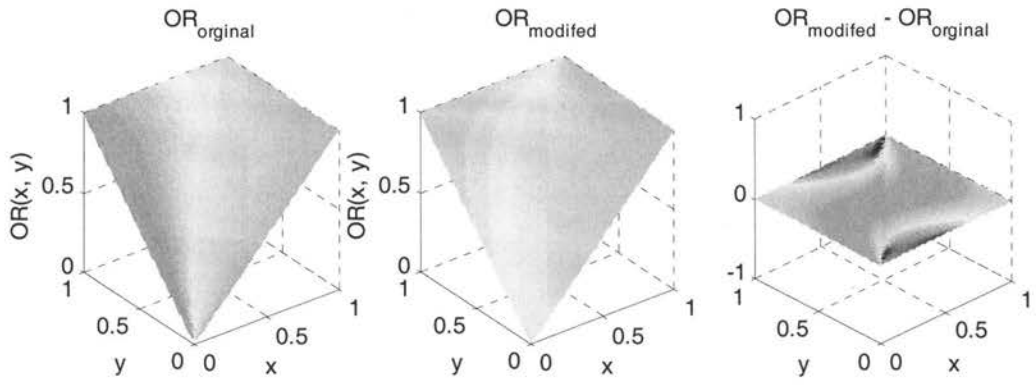


Figure 4.12 Comparison between original and modified sufficiency functions

Complementarity connective

Complementarity connective is a unary operator, representing the NOT relationship between the input and output. Traditionally it is defined as follows:

$$\mathrm{NOT}(x) = 1 - x. \tag{4.2.18}$$

Stover [29] proposed a new definition, which incorporates the ambiguity degree

$$\mathrm{NOT}(x) = \mathrm{AND}(1 - x, 1 - \alpha), \tag{4.2.19}$$

where α is the ambiguity degree.

We obey the traditional definition, but enhance it by two steps: introducing the significance degree and ambiguity degree (discussed in the next section **Ambiguity degree**) respectively.

Consider a complementarity input property with membership degree $x \in [0, 1]$ and significance degree $w \in [0, 1]$. We use $v(x, w) \in [0, 1]$ to denote the equivalent membership degree with unity significance degree. Membership degree should be preserved when significance degree is unity, and zero significance degree implies the property must be

ignored while computing the NOT of the input property. So $v(\bullet, \bullet)$ must satisfy the following boundary conditions:

$$v(x, 1) = x. \tag{4.2.20}$$

$$1 - v(x, 0) = 0 \Rightarrow v(x, 0) = 1. \tag{4.2.21}$$

(4.2.20) and (4.2.21) are the same as (4.2.1) and (4.2.2) of necessity connective. So it is easy to obtain the equivalent membership degree of complementarity connective according to that of necessity connective

$$v(x, w) = -w + wx + 1. \tag{4.2.22}$$

It is a general definition of membership degree, while in traditional definition the significance degree is set to one.

Ambiguity degree

Randomness in measurements

It is well established that there are two aspects of uncertainty, namely, vagueness and randomness. Vagueness may come from the uncertainty of our linguistic language and is handled by fuzzy logic theory, while randomness is due to the noise of data or error of our measurements. Probability and statistics provide a mathematical formalism to deal with it. Although the two aspects of uncertainty are independent, in many situations they co-exist. So a decision system, such as a classifier, must be designed to deal with both of them.

The membership degree is a measure of the vagueness inherent in the property being classified. Another measure is required for quantifying the randomness inherent in the properties being analyzed / classified. For this purpose, we suggest the measure called the

ambiguity degree, which reflects the degree to which our knowledge is incomplete. Like the membership degree, this measure also takes values in the unit interval. The ambiguity degree is monotonically related to the amount of randomness.

Each input property should be associated with its membership degree as well as its ambiguity degree. The ambiguity degree provides a measure of the input measurement variability caused by the randomness uncertainty (noise). This measurement variability can be translated into the membership degree variability by using the FUZZIFY function, as shown in Figure 4.13.

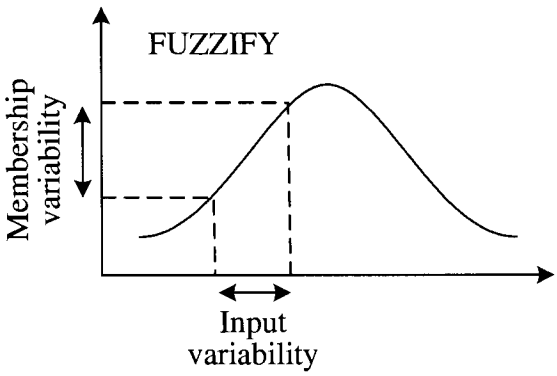


Figure 4.13 Membership variability caused by input variability

A noisy input value is distributed around the mean value according to some distribution, which is often specified by the mean and variance of measurements. We can choose them as the measurement of randomness and define the ambiguity degree accordingly. Variance can be looked as the power of noise and the square of mean can be regarded as the power of signal. Therefore, if the mean is not zero, the ambiguity degree $\alpha \in [0,1]$ of the input variable x is defined by

$$\alpha = \frac{\sigma_x^{\,2}}{\sigma_x^{\,2} + m_x^{\,2}}, \tag{4.3.1}$$

where m_x and σ_x^2 denote the mean and variance of x. Higher the variance, higher the ambiguity degree. If the variance is zero, the ambiguity degree is also zero, which means we have full confidence about the correctness of the input signal measurement.

Significance degree of the input in AND and OR connectives specifies the degree to which the corresponding input property is important to the output property. The ambiguity degree also affects the importance of input property to output property, higher the ambiguity, smaller its significance. So we can adjust the significance degree by ambiguity degree as follows:

$$w(\alpha) = w(1-\alpha) = w\frac{m_x^{\,2}}{\sigma_x^{\,2} + m_x^{\,2}}, \tag{4.3.2}$$

where w is the significance degree of x under no noise (zero variance), and $w(\alpha)$ is the modified significance degree taking ambiguity degree into account.

Calculation of ambiguity degree

Ambiguity degree is determined by the variance of input measurement, which is in turn decided by the distribution of input measurement. Therefore, for the input measurement, we need to know their probability density functions (pdf) to calculate ambiguity degree. In fuzzy aggregator the variance of input properties is translated into that of output property. Through operations of fuzzy aggregation, we can calculate the ambiguity degree of an output property after determining its pdf.

For this we first review some basic and useful probability knowledge as follows.

1. Let $f_X(x)$ denote the pdf of random variable x, m_x and σ_x^2 denote its mean and variance, and $E(\bullet)$ denote expectation function. Then m_x and σ_x^2 are calculated by

$$\begin{aligned} m_x &= E(X) = \int x f_X(x)dx, \\ \sigma_x^2 &= E(X^2) - [E(X)]^2 = \int x^2 f_X(x)dx - [\int x f_X(x)dx]^2. \end{aligned} \tag{4.3.3}$$

2. Suppose $y = g(x)$ is a monotonic function of random variable x and $f_X(x)$ is given, the pdf function of y, $f_Y(y)$, is calculated by

$$f_Y(y) = \left.\frac{f_X(x)}{|g'(x)|}\right|_{x=g^{-1}(y).} \tag{4.3.4}$$

$$\text{If } y = \frac{1}{x}, \text{ then } f_Y(y) = \frac{1}{y^2} f_X(\frac{1}{y}). \tag{4.3.5}$$

$$\text{If } y = x \pm 1, \text{ then } f_Y(y) = f_X(y \mp 1). \tag{4.3.6}$$

3. Suppose $z = x + y$ (x and y are independent), the pdf of z, $f_z(z)$ can be calculated by

$$f_Z(z) = \int f_X(x) f_Y(z-x)dx = \int f_X(z-y) f_Y(y)dy. \tag{4.3.7}$$

Based on these equations (from (4.3.3) to (4.3.7)), we proceed to calculate the ambiguity degree of the output variable z produced by AND and OR operations, given the pdf functions $f_X(x)$ and $f_Y(y)$ of input variables x and y.

Ambiguity degree of output of necessity connective: Our modified AND function is

$$\begin{aligned} z = \text{AND}(x, y) &= \frac{xy}{x + y - xy} \\ \Rightarrow \frac{1}{z} &= \frac{1}{x} + \frac{1}{y} - 1. \end{aligned} \tag{4.3.8}$$

Let us redefine $w = 1/z$, $u = 1/x$ and $v = 1/y$, we obtain from (4.3.8)

$$w = u + v - 1. \tag{4.3.9}$$

According to (4.3.5) and the new definitions of w, u and v, we have

$$\begin{aligned} f_U(u) &= \frac{1}{u^2} f_X(\frac{1}{u}), \\ f_V(v) &= \frac{1}{v^2} f_Y(\frac{1}{v}), \\ f_Z(z) &= \frac{1}{z^2} f_W(\frac{1}{z}). \end{aligned} \tag{4.3.10}$$

From (4.3.6), (4.3.7) and (4.3.9), we obtain

$$f_W(w) = \int f_U(u) f_V(w+1-u)du. \tag{4.3.11}$$

Combining (4.3.10) and (4.3.11), we obtain

$$\begin{aligned} f_Z(z) &= \frac{1}{z^2} \int f_U(u) f_V(\frac{1}{z}+1-u)du \\ &= \frac{1}{z^2} \int \frac{1}{u^2} f_X(\frac{1}{u}) \frac{1}{(\frac{1}{z}+1-u)^2} f_Y(\frac{1}{\frac{1}{z}+1-u})du. \end{aligned} \tag{4.3.12}$$

According to (4.3.3) and (4.3.12), we can find out the mean and variance of z by

$$\begin{aligned} m_z &= \int z f_Z(z)dz, \\ \sigma_z^2 &= \int z^2 f_Z(z)dz - [\int z f_Z(z)dz]^2. \end{aligned} \tag{4.3.13}$$

Then from (4.3.1) and (4.3.13), we can easily calculate the ambiguity degree of output property of necessity connective as follows:

$$\alpha = \frac{\sigma_z^2}{\sigma_z^2 + m_z^2}. \tag{4.3.14}$$

Ambiguity degree of output of sufficiency connective: Our modified OR function is

$$\begin{aligned} z &= \mathrm{OR}(x, y) = \frac{x+y-2xy}{1-xy} \\ \Rightarrow \frac{1}{z-1} &= \frac{1}{x-1} + \frac{1}{y-1} + 1. \end{aligned} \tag{4.3.15}$$

Let us redefine $w = 1/(z\text{-}1)$, $u = 1/(x\text{-}1)$ and $v = 1/(y\text{-}1)$, we obtain from (4.3.15)

$$w = u + v + 1. \tag{4.3.16}$$

According to (4.3.5), (4.3.6) and the new definitions of w, u and v, we have

$$\begin{aligned} f_U(u) &= \frac{1}{u^2} f_X(\frac{1}{u}+1), \\ f_V(v) &= \frac{1}{v^2} f_Y(\frac{1}{v}+1), \\ f_Z(z) &= \frac{1}{(z-1)^2} f_W(\frac{1}{z-1}). \end{aligned} \tag{4.3.17}$$

From (4.3.6), (4.3.7) and (4.3.16), we obtain

$$f_W(w) = \int f_U(u) f_V(w-1-u)du. \tag{4.3.18}$$

Combining (4.3.17) and (4.3.18), we obtain

$$\begin{aligned} f_Z(z) &= \frac{1}{(z-1)^2} \int f_U(u) f_V(\frac{1}{z-1}-1-u)du \\ &= \frac{1}{(z-1)^2} \int \frac{1}{u^2} f_X(\frac{1}{u}+1) \frac{1}{(\frac{1}{z}-1-u)^2} f_Y(\frac{1}{\frac{1}{z}-1-u}+1)du. \end{aligned} \tag{4.3.19}$$

According to (4.3.3) and (4.3.19), we can find out the mean and variance of z by

$$\begin{aligned} m_z &= \int z f_Z(z)dz, \\ \sigma_z^2 &= \int z^2 f_Z(z)dz - [\int z f_Z(z)dz]^2. \end{aligned} \tag{4.3.20}$$

Then from (4.3.1) and (4.3.20), we can calculate the ambiguity degree of output property of sufficiency connective as follows:

$$\alpha = \frac{\sigma_z^2}{\sigma_z^2 + m_z^2}. \tag{4.3.21}$$

Ambiguity degree of output of complementarity connective: Our NOT function is

$$y = \text{NOT}(x) = 1 - x. \tag{4.3.22}$$

Considering the significance degree and ambiguity degree together and based on (4.2.22), (4.3.2) and (4.3.22), we obtain

$$y = 1 - v(x, w(\alpha))$$
$$= 1 - (1 - w(1 - \alpha) + xw(1 - \alpha)). \tag{4.3.23}$$

If $w = 1$, we have

$$y = 1 - (\alpha + x - x\alpha)$$
$$= (1 - x)(1 - \alpha). \tag{4.3.24}$$

Incidentally, the output of complementarity connective derived above, incorporating the unit significance degree and ambiguity degree, is the same as that used by Stover [29] while choosing dot product as AND operation. Our derivation provides a justification to the usage of the formula in [29].

To calculate the ambiguity degree of the output, according to (4.3.4) and (4.3.22) we can easily obtain

$$f_Y(y) = f_X(1 - y). \tag{4.3.25}$$

According to (4.3.3) and (4.3.25), we can find out the variance of y by

$$\begin{aligned} m_y &= \int y f_Y(y) dy, \\ \sigma_y^2 &= \int y^2 f_Y(y) dy - [\int y f_Y(y) dy]^2. \end{aligned} \tag{4.3.26}$$

Then from (4.3.1) and (4.3.26), we can calculate the ambiguity degree of output property of complementarity connective as follows:

$$\alpha = \frac{\sigma_y^2}{\sigma_y^2 + m_y^2}. \tag{4.3.27}$$

Thus, based on the transform of the functions of necessity, sufficiency and complementarity connectives, we can determine the pdf function of output variable through

those of input variables, and then calculate the ambiguity degree of output property. The method for AND, OR and NOT can be combined to obtain variance of output property of a CINET classifier given variances of all input properties.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

CINET classifier is a kind of neuro-fuzzy classification system, which combines the distributed topology of neural networks and interpretability of if-then rules of fuzzy systems. Each CINET classifier represents a complicated fuzzy rule with arbitrary composition of fuzzy connectives such as AND, OR, and NOT. Based on distributivity law, the rule can be transformed as a disjunction of multiple conjunction sub-clauses. Each input variable of CINET classifier is associated with a membership function, which specifies the degree of existence of the corresponding input property. Sensory signal is submitted to CINET classifier and fuzzified to facilitate the inference within fuzzy aggregator. The output of CINET classifier is the membership degree specifying the existence of the interesting output property.

In this thesis some enhancements of CINET classifier have been proposed. First, we studied the problem of learning the range of input membership function of CINET classifier. We showed that the rule represented by CINET classifier could be decomposed as a union of multiple constraints. The points that satisfy a constraint form a hypercube in input space. Our goal is to minimize the total length of all edges of the hyper-cubes forming the union. We proved this to be a problem of Linear Mixed Integer Programming (LMIP), which can be solved by programming software such as ILOG CPLEX [7].

Then, the necessity and sufficiency functions in fuzzy aggregator were modified to satisfy some algebraic properties. We proved that the computation of significance degree in CINET classifier by Stover [29] is in fact a normalization process. Modified connective

functions are simpler and algebraically better behavioral, thus facilitating the use of back-propagation algorithm to adjust system parameters.

Finally we defined the ambiguity degree to deal with the randomness in input measurements. The calculation method of ambiguity degree is also provided by determining the pdf function of output from those of the inputs. Thus we combined fuzzy logic theory and probability theory together to deal with linguistic vagueness and measurement randomness.

Future work may be made in various aspects. We need experiment to verify that CPLEX is feasible to solve the problem of learning the range of input membership function. CINET classifier has been successfully applied to the control of underwater vehicles in the project conducted by Stover. We hope the data collected from that project can be used to verify our enhancements. This work will be done while I pursue my Ph.D.

Moreover, the effect of simplified connective functions on the performance of classification should also be studied through some simulation studies. We need to also examine the effect of randomness on the performance of the classifier following the approach presented above.

How one uses back-propagation to train the optimum weights (significance degrees) is an interesting research question. The output membership degrees are often hard to obtain when building training samples. Generally the samples only contain the class label (the output property), without numeric output membership degree. Can we still apply back-propagation algorithm to adjust system parameters? Lee [18] introduced a method, which adds a defuzzification neural network (DFNet) at the end of fuzzy inference part and uses this pre-learned DFNet to simulate any kind of defuzzification processes. The method can be used in CINET classifier to learn its maximum defuzzification process having multiple

output properties (for CINET classifier with only one output property, a sigmoid function suffices to simulate the step function for its defuzzification). We expect to test this method to tune system parameters by back-propagation algorithm.

The last direction is about the complexity of the CINET classifier. Based on the learning algorithms mentioned above, we hope to determine how many samples are sufficient to achieve desired classification performance. PAC learning theory [31] is useful in this direction.

BIBLIOGRAPHY

[1] Bentley, J. L., Multidimensional binary search trees used for associative searching, *Communications of the ACM*, Vol. 18, pp. 509-517, 1975.

[2] Bezdek, J. C., *Pattern recognition with fuzzy objective function algorithms*, Plenum Press, New York, 1981.

[3] Cherkassky, V., and Mulier, F., *Learning from data*, Wiley-Interscience, New York, 1998.

[4] Chiu, S. L., Fuzzy model identification based on clustering estimation, *Journal of Intelligent and Fuzzy Systems*, Vol. 2, No. 3, pp. 267-278, 1994.

[5] Funahashi, K. L., On the approximate realization of continuous mappings by neural networks, *Neural Networks*, Vol. 2, pp. 183-192, 1989.

[6] Horikawa, S., Furuhashi, T., and Uchikawa, T., On fuzzy modeling using fuzzy neural netoworks with the back-propagation algorithm, *IEEE Trans. Neural Networks.*, Vol. 3, No. 5, pp. 801-806, 1992.

[7] ILOG CPLEX, http://www.ilog.com/products/cplex/, March 27, 2003.

[8] Jang, J. S. R., ANFIS: adaptive-network-based fuzzy inference system, *IEEE Trans. Systems Man and Cybernetics*, Vol. 23, No. 3, pp. 665-685, 1993.

[9] Jang, J. S. R., Sun, C. T., Neuro-fuzzy modeling and control, *IEEE proceedings*, Vol. 83, No. 3, pp. 378-406, 1995.

[10] Janikow, C. Z., Fuzzy decision tree: issues and methods, *IEEE Trans. Systems Man and Cybernetics*, Vol. 28, No. 1, pp. 1-14, 1998.

[11] Karr, C. L., and Gentry, E. J., Fuzzy control of pH using genetic algorithm, *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 1, pp. 46-53, 1993.

[12] Kearns, M. J., and Vazirani, U. V., *An introduction to computational learning theory*, MIT Press, Cambridge, Massachusetts, 1994.

[13] Kosko, B., Fuzzy system as universal approximations, *IEEE Trans. Computers*, Vol. 43, No. 11, pp. 1329-1333, 1994.

[14] Kumar, R., and Stover, J. A., The CINET fuzzy classifier: formal background and enhancements, *Proceedings of IEEE International Symposium on Intelligent Control / Intelligent Systems and Semiotics*, pp. 314-319, 1999.

[15] Kuncheva, L. I., *Fuzzy classifier design*, Physica-Verlag, Heidelberg, 2000.

[16] Lee, C. C., Fuzzy logic in control systems: Fuzzy logic controller, Part I, *IEEE Trans. Systems Man and Cybernetics*, Vol. 20, No. 2, pp. 404-418, 1990.

[17] Lee, C. C., Fuzzy logic in control systems: Fuzzy logic controller, Part II, *IEEE Trans. Systems Man and Cybernetics*, Vol. 20, No. 2, pp. 419-435, 1990.

[18] Lee, K. M., Kwak, D. H, and Hyung, L. K, Fuzzy inference neural network for fuzzy model tuning, *IEEE Trans. Systems Man and Cybernetics*, Vol. 26, No. 6, pp. 637-645, 1996.

[19] Lee, S. C., and Lee, E. T., Fuzzy neural networks, *Int. Journal on Math. Biosciences*, Vol. 23, pp. 151-177, 1975.

[20] Lin, C. T., and Lee, C. S. G., Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Computers*, Vol. 40, No. 12, pp. 1320-1336, 1991.

[21] Mamdani, E. H., and Assilian, S., An experiment in linguistic synthesis with a fuzzy logic controller, *Int. Journal on Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.

[22] Nauck, D., and Kruse R., *http://fuzzy.cs.uni-magdeburg.de/nfdef.html*, March 13, 2003.

[23] Roubos, H., and Setnes, M., Compact fuzzy models through complexity reduction and evolutionary optimization, *In Proc. of IEEE international conference on fuzzy systems*, San Antonio, USA, pp. 762-767, 2000.

[24] Russel, S., and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, New York, 2000.

[25] Setnes, M., and Roubos, H., GA-fuzzy modeling and classification: complexity and performance, *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 5, pp. 509 -522, 2000.

[26] Setnes, M., Babuska, R., Kaymak, U. etc., Similarity measures in fuzzy rule base simplification Systems, *IEEE Trans. Systems Man and Cybernetics*, Vol. 28, No. 3, pp. 376 - 386, 1998.

[27] Setnes, M., Babuska, R., Verbruggen, H. B., Rule-based modeling: precision and transparency, *IEEE Trans. Systems Man and Cybernetics*, Vol. 28, No. 1, pp. 165 -169, 1998.

[28] Sun, C. T., Rule-based structure identification in an adaptive-network-based fuzzy inference system, *IEEE Trans. Fuzzy Systems*, Vol. 2, No. 1, pp. 64-73, 1994.

[29] Stover, J. A., Hall, D. L., and Gibson, R. E., A fuzzy-logic architecture for autonomous multisensor data fusion, *IEEE Trans. Industrial Electronics*, Vol. 43, No. 3, pp. 403-410, 1996.

[30] Takagi, T., and Sugeno, M., Fuzzy identification of systems and its application to modeling and control, *IEEE Trans. Systems Man and Cybernetics*, Vol. 15, No. 1, pp. 116-132, 1985.

[31] Vidyasagar, M., *A theory of learning and generalization: with applications to neural networks and control systems,* Springer, New York, 1997.

[32] Wang, L. X., and Mendal, J. M., Fuzzy basis function, universal approximation, and orthogonal least-squares learning, *IEEE Trans. Neural Networks.*, Vol. 3, No. 5, pp. 807-814, 1992.

[33] Wang, L. X., and Mendal, J. M., Generating fuzzy rules by learning from examples, *IEEE Trans. Systems Man and Cybernetics.*, Vol. 22, No. 6, pp. 1414-1427, 1992.

[34] Wang, L. X., and Mendal, J. M., Back-propagation fuzzy system as nonlinear dynamic system identifiers, *In Proc. of IEEE international conference on fuzzy systems*, pp. 1409-1416, 1992.

[35] Yager, R. R., and Filev, D. P., Generation of fuzzy rules by mountain clustering, *Journal of Intelligent and Fuzzy Systems*, Vol. 2, No. 3, pp. 209-219, 1994.